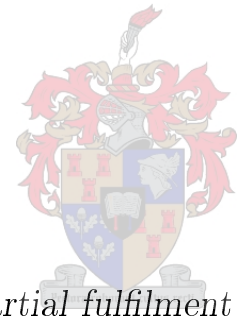
Image deblurring and blur kernel estimation of motion blurred colour photographs

by

Jan-at Engelbrecht



Thesis presented in partial fulfilment of the requirements for the degree of Master of Science (Applied Mathematics) in the Faculty of Natural Sciences at Stellenbosch University

Supervisor: Dr. M.F. Maritz

March 2016

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: March 2016

Abstract

Image deblurring and blur kernel estimation of motion blurred colour photographs

J. Engelbrecht

Department of Mathematical Sciences,
Division of Applied Mathematics,
University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa.

Thesis: MSc

February 2016

Blurred images can be restored using known deconvolution methods. Image blur can be caused by various factors during the image acquisition phase, such as camera defocus or motion. The restoration of blurred images is further complicated by the addition of possible electronic noise.
In this report we propose a direct method where the kernel is estimated by finding possible edges in the blurred image using edge detection. A sinogram of the blur kernel is obtained by sampling and then differentiating across edges in the image for a sufficient number of edge orientations. This sinogram is then improved by making use of a sinogram interpolation technique. The application of the inverse Radon transform then yields the kernel which is used to restore the image.

Uittreksel

Beeldontvaging en beraming van die vervagingskern van bewegings-vervaagde kleurfotos fotos

("Image deblurring and blur kernel estimation of motion blurred colour photographs")

J. Engelbrecht

Departement Wiskundige wetenskappe,
Afdeling Toegepaste Wiskunde,
Universiteit van Stellenbosch,
Privaatsak X1, Matieland 7602, Suid Afrika.

Tesis: MSc

Februarie 2016

Vervaagde (d.w.s. uit-fokus) beelde kan herstel word met behulp van bekende dekonvolusiemetodes. Beeldvervaging kan deur verskeie faktore gedurende die verkryging van die beeld veroorsaak word, soos swak fokus of deur beweging van die kamera. Die herstel van vervaagde beelde word verder gekompliseer deur die toevoeging van moontlike elektroniese ruis.

In hierdie tesisword 'n direkte metode voorgestel, waar die vervagingskern beraam word deur moontlike rande in die vervaagde beeld met behulp van die randopsporingstegnieke te vind. 'n Sinogram van die vervagingskern word dan verkry deur oor verskeie rande in die beeld te monster vir 'n voldoende aantal randoriÃ«ntasies en dan te differensieer. Die sinogram word dan verbeter deur gebruik te maak van 'n sinogram-interpolasietegniek. Die toepassing van die inverse Radon-transform verskaf die kern, wat gebruik word om die beeld te herstel.

Acknowledgements

I would like to express my sincere gratitude to the following people. My parents and family for supporting me throughout my academic career. My friends who were always close by with a helpful word and encouragement. My supervisor Dr MF Maritz who has guided and supported me throughout the completion of this document with patience.

Dedications

Hierdie tesis word opgedra aan my familie.

Contents

List of Figures

Notation and important symbols

Notation

A	All capitalized letters are matrices.
a	All uncapitalized letters are constants or variables.
$\vec{a}$	All arrowed and uncapitalized letters are vectors.
$\hat{A}$	All hatted letters are Fourier domain elements.
$\check{i}$	All checked letters are set, matrix or vector sizes.
$\bar{K}$	Set complex conjugate or average.

Special variables, vectors, matrices, and functions

e^x	The exponential function.
ϵ	The expected error.
$\varepsilon(x)$	Expected value of x.
$L_l(x)$	Log likelihood of x.
$N_l(x)$	Negative log likelihood of x.
N	The normal distribution.
E	The exponential distribution.
μ	The mean.
Σ	The sum when subscripted, otherwise the covariance matrix.
$\mathfrak{F}$	The Fourier transform.
$\mathfrak{R}$	The Radon transform.

Subscripts

i, j	Matrix row and column indices.
x, y	Continuous image coordinates.

Chapter 1

Introduction and problem statement

1.1 Introduction

When a photograph is taken, light is captured from the real world scene or simply the scene. In the case of digital images, a photograph is stored as a matrix where each element is called a pixel, with a value between 0 and 255. A pixel records the intensity value of the light[1] that was captured from a small region in the scene during the camera lens exposure period. When the camera is moved during the exposure period the photograph experiences what is known as *motion blur*. The movement of the camera smears the light that is captured, for each pixel, along a motion path. The result is a motion blurred digital image.

Figure 1.1: Typical image blur, lens defocus and motion blur.

The blurred image model

[1] The amount of light reflected or emitted by any object or person.

A digital image is stored as a matrix of pixels. A blurred image $B(x, y)$ is modeled by the convolution of a sharp image $A(x, y)$ with a blur kernel $K(x, y)$ plus noise $N(x, y)$,

$$B = A * K + N.$$

It will be shown that a blurred image can accurately be reconstructed using deconvolution methods given that the original blur kernel is known. The most important deconvolution techniques are those of Lucy (1974) and Weiner (1949).

However, when working with blurred photographs the original blur kernel is often completely unknown. The problem is that the original true blur kernel can never be fully recovered from a blurred photograph, only estimated. Deconvolution of a blurred image with incorrect blur kernels results in noise and ringing artifacts in the deblurred image. The blurred image problem is extremely ill conditioned however, research into *image deblurring*[2], *blur kernel estimation*[3] and *blind deconvolution*[4] has been shown to be successful under various circumstances.

We will consider the work done by Fegus *et al.* (2006), to be a bench mark for kernel estimation algorithms that have been produced since 2006, since it has been cited by all of the following articles Jia (2007); Levin *et al.* (2009); Levin (2006); Joshi *et al.* (2008); Cho *et al.* (2011); Qi Shan and Jia (2007); Schuon and Diepold (2009) and more.

These blur kernel estimation and deblurring algorithms include a number of different methods for motion blur estimation, based on natural image statistics, for example Fegus *et al.* (2006) and edge based parameters estimation , for example Joshi *et al.* (2008); Cho *et al.* (2011). Although these techniques may work well for a wide range of images and blur kernels, they make use of maximum a-posteriori estimation (MAP) techniques to estimate the blur kernel. It has however been pointed out that MAP estimation may not be ideal for the purpose of blur kernel identification, Levin *et al.* (2009) and that it has disadvantages to be addressed. The disadvantages that MAP estimation can incur include, the requirement of a large parameter space, the need for a good initial guess and the possibility that MAP estimation may often suffer from high computational cost.

The Radon transform method first described by Sun *et al.* (2009) and later

[2] Deconvolution of a blurred image with an estimated blur kernel.

[3] Estimation of an image blur kernel often also referred to as point spread function estimation

[4] Recovering the sharp version of a blurred image using deconvolution when the blur kernel is unknown.

improved upon by Cho *et al.* (2011) to include estimation of non-linear motion blur kernels, presented the opportunity to create a direct method. However, both of these algorithms still include a MAP estimation procedure to produce the desired kernel.

Nevertheless the proposed direct inversion method, which is done by finding possible edges in the blurred image via edge detection, allows us to obtain a sinogram of the blur kernel by sampling and then differentiating across edges in the image for a sufficient number of edge orientations. In this thesis a technique for improving the proposed tomographic blur kernel estimation method via sinogram interpolation will be presented. Application and research done in this field will be briefly discussed. A method for sinogram interpolation based on the work done by Kalke and Siltanen (2014) and its application to the tomographic algorithm is described.

It will be shown that various motion blur kernels can be simulated, which in turn allows for the simulation of motion blurred images. Simulated and real blurred photographs will be used to show that the blur kernel can indeed be estimated form a single blurred image. The theory and implementation are discussed and typical results are shown.

1.2 Objectives

The main objective of this study is to explore and implement a direct method of blur kernel estimation for blind deconvolution of motion blurred images. The edge-based Radon transform method described by Cho *et al.* (2011) is used as starting point. The method presented in this thesis will extract edges from the blurred image, from which a partial sinogram of the kernel is assembled by sampling along the edge normal. The kernel is obtained by applying the inverse Radon transform to the sinogram.

The aim is to improve on existing kernel estimation methods so that reasonable kernel estimation for non-linear motion blur can be performed. In this thesis, focus is placed on implementing and improving edge extraction techniques and sampling methods as these lead to a more accurate sampled sinogram. Thereafter, the aim is to improve the obtained sinogram by employing a sinogram interpolation technique. The overall objective of this study is to implement a complete system for kernel estimation, to illustrate its capabilities with simulated kernels and to evaluate its performance for motion blurred photographs. It should be noted that only two main assumption were made about the blurred photographs that are dealt with in this thesis; they are as follows:

- The photographs and images that are dealt with are all translation mo-

tion or aperture blurred. Blur occurring from camera rotation is not considered.

- For all images and photographs that are affected by blur, the blur is spatially invariant. This means that, the whole image was affected by the same blur kernel and motion.

Various methods of kernel estimation were studied, specifically the natural image statistic based method presented by Fegus *et al.* (2006), the sharp edge prediction method presented by Joshi *et al.* (2008) and the edge-based Radon transform method described by Cho *et al.* (2011). Relevant theory is covered from these studies in order to draw an objective comparison between various blur kernel estimation techniques.

1.3 Layout of document

This document will follow the following layout:
Chapters $(1-3)$ will introduce the blurred image problem and provide an overall description of the research that has been done in the field. Specifically in Chapter (2) methods will presented, by which motion and aperture blur can be simulated, then briefly show how a blurred image can be restored using deconvolution. Chapter (3) will provide an overview of important and current blind deconvolution methods. This will conclude the introductory chapters.
Chapter $(4-8)$, will cover the work that was done during the thesis, with the aim on producing a working image deblurring algorithm. Chapter (4) will present the mathematical theory which forms the foundation of the tomographic estimation algorithm. Chapter (5) will cover the various edge based algorithms needed to complete a practical implementation of the algorithm. Chapter (6) will illustrate the sparse inversion problem that is encountered by the tomographic estimation algorithm and present a sinogram interpolation technique, with which the problem is solved. Lastly, chapters $(7-8)$ will present obtained results and discuss the tomographic estimation algorithm as a practical image deblurring algorithm.

Chapter 2

Deconvolution and modeling image blur

2.1 Modeling image blur

A blurred image B is modeled as the convolution of *sharp image* A with a degradation function, called the *kernel* K, plus *noise* N, such that

$$B(x, y) = A(x, y) * K(x, y) + N(x, y), \tag{2.1.1}$$

Here $*$ denotes two dimensional convolution. For the duration of this chapter we will assume that no noise has been added to the blurred image.
Blur occurs while the photograph is being taken. Since taking a photograph is a process that happens over time, it is concluded that the blur is also captured over time. To be exact, the image and the blur is captured in the length of time a shutter of the camera is open, called the *exposure time*[1].

A distinction is made between types of image blur. The blur that occurs due to a lens system being out of focus is called *aperture blur*. *Motion blur* occurs when a camera is moved during the image capturing process. This often creates a path or trail along which the image is smeared. Motion blur that occurs in a single direction will be referred to as linear motion blur or simply *linear blur*. The term *non-linear blur* will be used to refer to image blur caused by the blur kernel having more than one mode in any direction.

2.2 Simulating blur kernels

Equation (2.1.1) can be used to simulate a blurred image B, however this can only be done if a suitable blur kernel K is available.

[1] The amount of light that reaches the film or image sensor is proportional to the exposure time.

Two main assumptions are made about motion blur kernels throughout this thesis. The first is that the blur kernel B is always positive. Secondly, it is assumed that a blur kernel is energy conserving, that is

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} K(x,y)dxdy = 1. \tag{2.2.1}$$

This means that a kernel does not affect the mean image intensity when a photograph is taken. This condition can always be satisfied by normalizing the kernel K, before applying it to a sharp image A.

2.2.1 Linear and aperture simulation using a Gaussian function

In order to simulate blur kernels, a version of the Gaussian function will be used. The multi-variate Gaussian function is given by

$$f_X(x_1,\ldots,x_k) = \frac{1}{\sqrt{2\pi^k|\Sigma|}} e^{-\frac{1}{2}(X-\mu)^t\Sigma^{-1}(X-\mu)}. \tag{2.2.2}$$

The covariance matrix Σ is symmetric positive definite and μ is the center position of the distribution, where X ia a k dimensional column vector.
The following function P, see Figure 2.2.1, will be used to simulate aperture and linear blur kernels at an angle θ, such that

$$P(x,y) = e^{-(a(x-\mu_x)^2+2b(x-\mu_x)(y-\mu_y)+c(y-\mu_y)^2)}.$$

The constants a, b and c are calculated as follows,

$$a = \frac{\cos^2(\theta)}{2\sigma_x^2} + \frac{\sin^2(\theta)}{2\sigma_y^2}$$

$$b = -\frac{\sin(2\theta)}{4\sigma_x^2} + \frac{\sin(2\theta)}{4\sigma_y^2}$$

$$c = \frac{\sin^2(\theta)}{2\sigma_x^2} + \frac{\cos^2(\theta)}{2\sigma_y^2}$$

Normalizing P yields the simulated blur kernel K,

$$K(x,y) = P(x,y)/\left[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} P(x,y)dxdy\right].$$

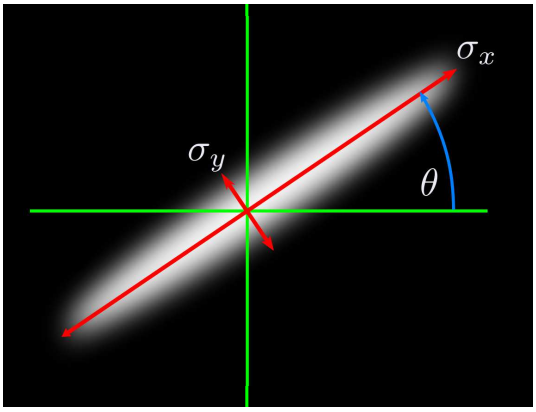
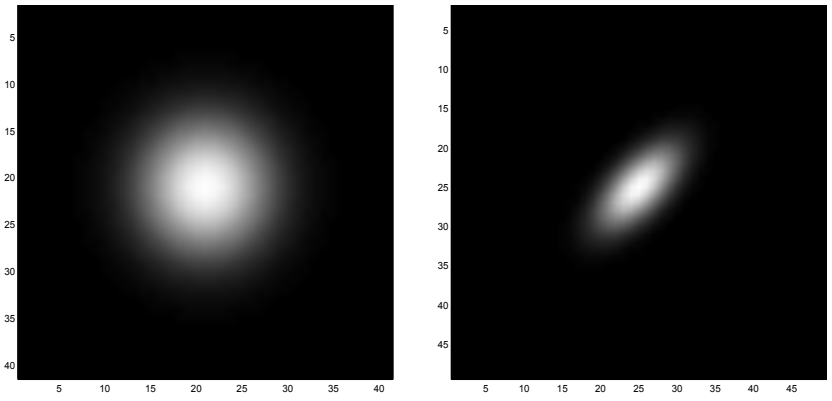


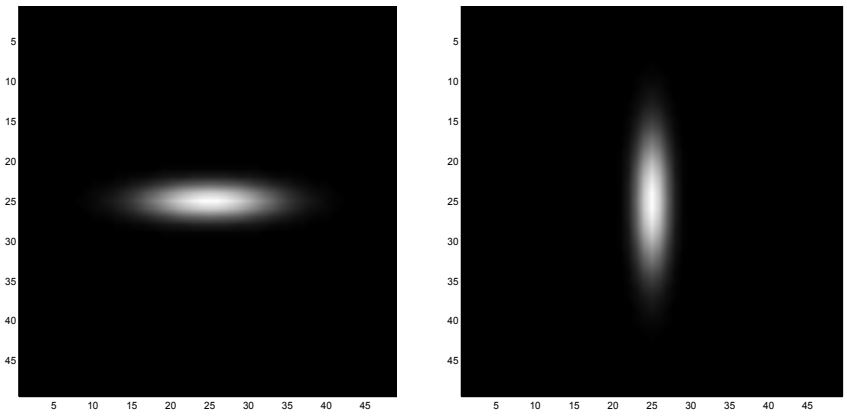
Figure 2.1: Linear Gaussian blur kernel.

This allows the creation of blur kernels without having to take the time based nature of the kernel into account. Some examples of blur kernels created with the Gaussian function are given below:



(A1) (A2)

Figure 2.2: A circular kernel (A1), ($\sigma_x = \sigma_y$) and a linear motion kernel (A2) at an angle $\theta > 0, (\sigma_x < \sigma_y)$.



(A1) (A2)

Figure 2.3: Vertical (A1) or horizontal motion (A2), that is when $\theta = 0, (A1)(\sigma_x < \sigma_y)$, or $(\sigma_x > \sigma_y)$.

Although this method of blur kernel creation is simple and time efficient, it does not allow the creation of non-linear blur kernels.

2.3 Motion blur

In order to simulate a photograph that has been affected by motion blur, a mathematical model can be derived from first principles, see Gonzalez and Woods (2008).
Suppose that a sharp image $A(x, y)$ has been blurred by some motion $[x(t), y(t)]$ $t \in [0, T]$ for a duration of time T. The resulting blurred image $B(x, y)$ is given by

$$B(x, y) = \frac{1}{T} \int_0^T A(x - x(t), y - y(t))dt.$$

Taking the Fourier transform of $B(x, y)$ yields

$$\begin{aligned} \hat{B}(u, v) &= \frac{1}{T} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} [B(x, y)]\, e^{-2\pi i(ux+vy)} dxdy, \\ &= \frac{1}{T} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \left[\int_0^T A(x - x(t), y - y(t)dt \right] e^{-2\pi i(ux+vy)} dxdy. \end{aligned} \tag{2.3.1}$$

Reversing the order of integration, the Fourier transform of the blurred image $B(x, y)$ is obtained, as follows

$$\begin{aligned} \hat{B}(u, v) &= \frac{1}{T} \int_0^T \left[\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} A(x - x(t), y - y(t)) e^{-2\pi i(ux+vy)} dxdy \right] dt, \\ &= \frac{1}{T} \int_0^T \hat{A}(x, y) e^{-2\pi i(ux(t)+vy(t))} dt, \end{aligned}$$

$$= \frac{\hat{A}(x,y)}{T} \int_0^T e^{-2\pi i(ux(t)+vy(t))} dt. \tag{2.3.2}$$

This allows us to define the blur kernel K that has degraded $A(x,y)$ as

$$K(x,y) = \mathfrak{F}^{-1}\left\{ \frac{1}{T} \int_0^T e^{-2\pi i(ux(t)+vy(t))} dt \right\}.$$

The inverse Fourier transform of (2.3.2) then yields the desired model

$$B(x,y) = A(x,y) * K(x,y).$$

This equation contains a path following blur kernel. The result is expected, but helps to explain the creation of most blur kernels. The kernel creation time simply is the exposure time of determined by the opening and closing of the camera lens.

2.4 Simulating motion blur

Motion blur can be simulated by moving a circular Gaussian function P along a path[2] $C := [c_x(t), c_y(t)]$ $t \in [0,T]$. The resulting blur kernel K is given by

$$K(x,y) = \frac{1}{T} \int_0^T \frac{P(x - c_x(t), y - c_y(t))}{1 + \sqrt{\left(dc_x/dt\right)^2 + \left(dc_y/dt\right)^2}} dt. \tag{2.4.1}$$

This provides an interesting interpretation for the occurrence of lighter and darker areas in a motion kernel. The expression

$$V(t) = \sqrt{\left(\frac{dc_x}{dt}\right)^2 + \left(\frac{dc_y}{dt}\right)^2},$$

can be seen as the speed at which the camera is moved while taking the photograph.
Assume that a kernel $K(x,y)$ is formed at constant speed c, as follows,

$$K(x,y) = \frac{1}{T} \int_0^T \frac{P(x - k_1 t, y - k_2 t)}{c} dt \tag{2.4.2}$$

$$= \frac{1}{cT} \int_0^T P(x - k_1 t, y - k_2 t) dt \tag{2.4.3}$$

$$= \frac{1}{cT} (P(x - k_1 T, y - k_2 T) - P(x,y)). \tag{2.4.4}$$

[2]Here a Gaussian function is used, but P can be replaced by any circular function that best models the specific parameters of a camera lens.

This implies that the intensity at a point $[x_1, y_1]$ in $K(x, y)$ is weighted by $1/c$. Therefore, the intensity ρ at a point in a $K(x, y)$ is proportional to the time spent at that point,

$$\rho \propto \frac{1}{V(t)}.$$

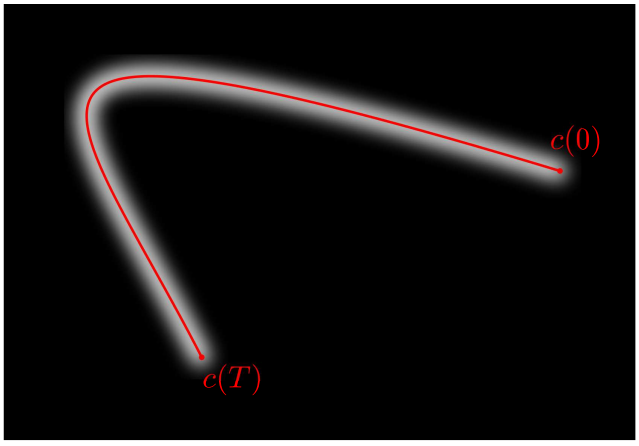


Figure 2.4: Motion blur at a constant rate over time.

If the speed is low during a period, there will be a brighter area in the kernel. When the camera is moved quickly, lower grey values area produced. However, while simulating it may be easier to replace $1/V(t)$ with an intensity function $\rho(t)$. This is interpreted as a camera that moved at constant speed while the aperture intensity ρ is changed over time. This allows us to restructure (2.4.1) into an integral consisting of two independent functions,

$$K'(x, y) = \int_0^T \rho(t) P(x - c_x(t), y - c_y(t)) dt. \tag{2.4.5}$$

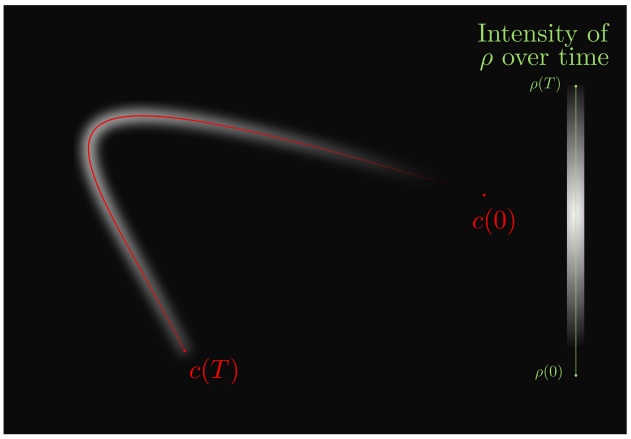


Figure 2.5: Motion blur over time with ρ increasing over time.

2.5 The discrete blur model

The blur model can further be extended by mapping the path $C = \{(c_x(t), c_y(t))|t \in [0, T]\}$ onto a discrete grid. The convolution of the discrete path and a general probability distribution $P(x, y)$ would then produce the blur kernel. The continuous domain and path can be discretized as follows:

$$x_i = i\Delta x, \;\; y_j = j\Delta y, \;\; i = 0, \ldots, \check{i}, \;\; j = 0, \ldots, \check{j}$$

$$t_m = m\Delta t, \;\; \vec{c_m} = (c_x(t_m), c_y(t_m)), \;\; m = 0, \ldots, \check{m}.$$

Having discretized the continuous domain and path, a function $S(x_i, y_j)$ is defined to determine the intensity at (x_i, y_j), such that

$$S(x_i, y_j) = \sum_{n=0}^{\check{m}} \frac{\delta(c_x(t_n) - x_i, \, c_y(t_n) - y_j)}{\check{m}}$$

with

$$\delta(a, b) = \begin{cases} 1 & \text{if } |a| < \Delta x/2, \text{ and } |b| < \Delta y/2 \\ 0 & \text{otherwise} \end{cases}.$$

This definition allows for a unique interpretation of the blur kernel. The kernel can be seen as the combination of a path S and a circular aperture function P. S is created by the camera motion and has intensity $\rho(t)$ at time t. This path is convolved with a *basis kernel* P, which describes the point distribution created by the camera lens. This is included into the blurred image model as follows

$$B(x, y) = A(x, y) * (S(x, y) * P(x, y)).$$

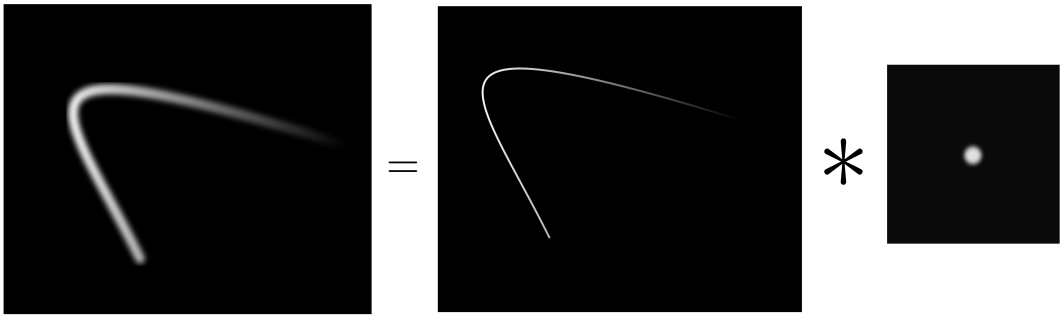


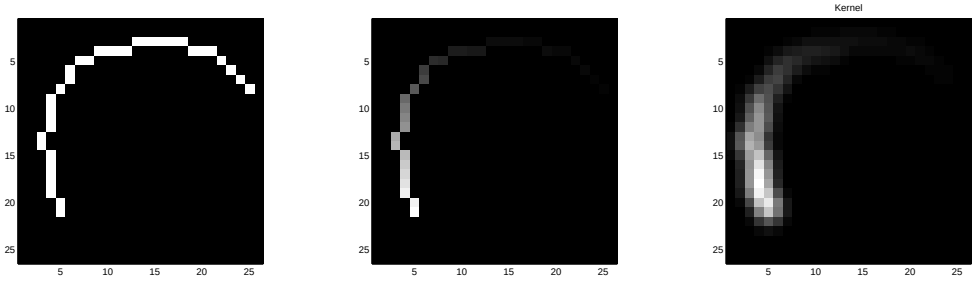
Figure 2.6: Convolution of path with a density function.

2.6 Simulation results

Some results are shown that were obtained by simulating motion blur using the discrete blur model. This method can be used to produce most blur kernels. The kernels are obtained by selecting points on a clear image to create a path. The points that lie along the path are then packed into an array of zeros, as ones. The intensity of the points are then altered using an intensity function, which creates an *intensity path.* The intensity path is then convolved with some circular two dimensional Gaussian function. The resulting function is then normalized to yield the kernel. Lastly the kernel is used to blur a test image.

For the following blur examples 1 and 2, random paths with random intensities were created. Two paths were convolved with a Gaussian function where $\sigma_1 = 1$ and $\sigma_2 = 3$ respectively.

Example 1:

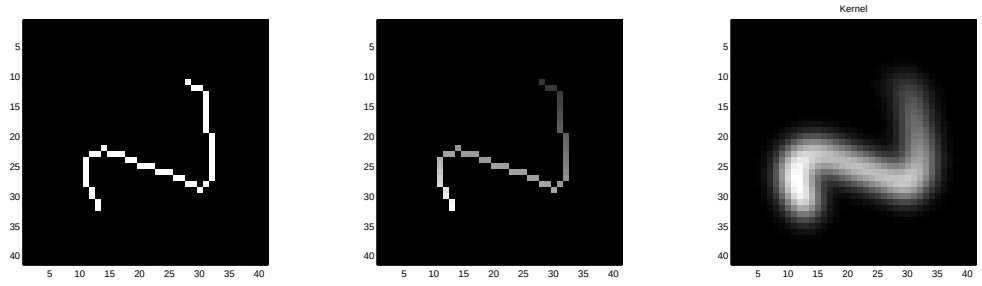


Motion path Intensity path Blur kernel

Original image Blurred image

Figure 2.7: Curved motion blur simulation.

Example 2:



Motion path Intensity path Blur kernel

Original image Blurred image

Figure 2.8: Extreme motion blur simulation.

The examples show that the simulation of intricate and intensity varying blur kernels, can regularly be produced by utilizing the discrete blur model. Moreover these blur kernels can then be used to simulate blurred images. This allows for testing to be done on simulated blurred image where the ground truth blur kernel is known.

2.7 Deconvolution

The restoration of motion blurred images involves correctly estimating the blur kernel K and then deconvolving the blurred image B with K to yield the restored image. In general, when deconvolving it is required to find a solution to the equation $K * A = B$, with the possibility of added noise N. Although this is an area that has been thoroughly researched, it remains important to investigate deconvolution methods, since it provides the basis on which deblurring is built.

2.7.1 Inverse filtering

The simplest method for image deconvolution is by direct inverse filtering. This is done by dividing the Fourier transform of the blurred image $\hat{B}(u, v)$ by the transform of the kernel $\hat{K}(u, v)$ within the frequency domain,

$$\hat{A}(u, v) = \frac{\hat{B}(u, v)}{\hat{K}(u, v)}.$$

However this can be problematic, since zero values often appear in the kernel transform. The values close to zero are therefore limited or cut-off, since they produce extreme values when $\hat{B}(u, v)/\hat{K}(u, v)$ is calculated. This minimizes the singular errors that are found in the restored image.

2.7.2 Weiner filtering

Weiner filtering works by attempting to minimize the impact of noise on the deconvolution process. Consider an image $\hat{A}(u, v)$ with added noise $\hat{N}(u, v)$, as random variables. The aim of Weiner filtering is to minimize the expected mean square error between the estimated original image $\hat{\hat{A}}$ and true image $\hat{A}$, that is the error

$$\epsilon = \varepsilon[(\hat{A} - \hat{\hat{A}})^2].$$

Assuming that the noise and ground truth image are independent, it is left to find a function $\hat{G}$ such that $\grave{\hat{A}} = \hat{G}B$. Substituting $\grave{\hat{A}} = \hat{G}\hat{B}$,

$$\begin{aligned}
\epsilon &= \varepsilon[(\hat{A} - \grave{\hat{A}})^2] \\
&= \varepsilon[(\hat{A} - \hat{G}\hat{B})^2], \\
&= \varepsilon[(\hat{A} - \hat{G}(\hat{A}\hat{K} + \hat{N}))^2], \\
&= \varepsilon[(\hat{A}(1 - \hat{G}\hat{K}) + \hat{G}\hat{N}))^2], \\
&= (1 - \hat{G}\hat{K})\overline{(1 - \hat{G}\hat{K})}\varepsilon[\hat{A}^2] - (1 - G\hat{K})\bar{G}\varepsilon[\hat{A}\bar{\hat{N}}] \\
&\quad - \overline{(1 - G\hat{K})}G\varepsilon[\hat{N}\bar{\hat{A}}] + \overline{(1 - \hat{G}\hat{K})}\hat{G}\varepsilon[\hat{N}\bar{\hat{A}}] + \hat{G}\bar{\hat{G}}\varepsilon[\hat{N}^2].
\end{aligned}$$

Now since the ground truth image and noise are independent, it can be shown that $\varepsilon[\hat{A}\bar{\hat{N}}] = 0$ $\varepsilon[\hat{A}\bar{\hat{N}}] = 0$ and $\varepsilon[\hat{N}\bar{\hat{A}}] = 0$. Along with the substitutions

$$\varepsilon[\hat{N}^2] = |\hat{N}|^2 = n_P \text{ and } \varepsilon[\hat{A}^2] = |\hat{A}|^2 = s_P,$$

the error can be rewritten as follows:

$$\epsilon = (1 - \hat{G}\hat{K})\overline{(1 - \hat{G}\hat{K})}s_p + \hat{G}\bar{\hat{G}}n_p.$$

What is left is to minimize the error function with respect to $\hat{G}$. This is done by taking the derivative of ε with respects to $\hat{G}$ and setting it equal to zero. It can then be shown that

$$\begin{aligned}
\frac{d\epsilon}{d\hat{G}} = \bar{\hat{G}}n_p - \hat{K}\overline{(1 - \hat{G} * \hat{K})}s_p = 0, \\
\Rightarrow \bar{\hat{G}}n_p = \hat{K}\overline{(1 - \hat{G}\hat{K})}s_p, \\
\Rightarrow \frac{\bar{\hat{G}}}{\overline{(1 - \hat{G}\hat{K})}} = \frac{\hat{K}s_p}{n_p}, \\
\Rightarrow \frac{\overline{(1 - \hat{G}\hat{K})}}{\bar{\hat{G}}} = \frac{n_p}{\hat{K}s_p}, \\
\Rightarrow \frac{\hat{K}}{\bar{\hat{G}}} - \frac{\hat{K}\overline{\hat{G}\hat{K}}}{\bar{\hat{G}}} = \frac{n_p}{s_p}, \\
\Rightarrow \frac{1}{\bar{\hat{G}}} = \frac{1}{\hat{K}}\left[\frac{n_p}{s_p} + |\hat{K}|^2\right], \\
\Rightarrow \bar{\hat{G}} = \frac{1}{\hat{K}}\left[\frac{|\hat{K}|^2}{n_p/s_p + |\hat{K}|^2}\right].
\end{aligned}$$

Therefore

$$\hat{\hat{A}} = \frac{1}{\hat{K}} \left[\frac{|\hat{K}|^2}{|\hat{K}|^2 + n_p/s_p} \right] B, \tag{2.7.1}$$

where s_p and n_p are the respective power spectrum of the ground truth image and the image noise. Further note that n_p/s_p is the reciprocal of the signal to noise ratio (SNR) of the blurred image. However since the power spectrum of the ground truth image often remains unknown, the SNR may simply be estimate by some constant k such that SNR $= 1/k$. Equation (2.7.1) can then be expressed as

$$\tilde{\hat{A}} = \frac{1}{\hat{K}} \left[\frac{|\hat{K}|^2}{|\hat{K}|^2 + k} \right] B. \tag{2.7.2}$$

This is a well known form of the Weiner filter for image deconvolution.

2.7.3 Lucy-Richardson deconvolution

Lucy-Richardson deconvolution is an iterative procedure for restoring blurred images. Unlike Weiner filtering, the Lucy-Richardson algorithm is based on Bayesian methods to produce the restored image. When given a 1-dimensional image formation model $B = A * K$, Bayes's theorem can be applied to the conditional probability $P(A|B)$ (since A and B are dependant via K) such that,

$$P(A_i|B_h) = \frac{P(B_h|A_i)P(A_i)}{\sum_j P(B_h|A_j)P(A_j)}, \tag{2.7.3}$$

where

$$i \in \{1, ..., \check{i}\},\ j \in \{1, ..., \check{j}\} \text{ and } h \in \{1, ..., \check{h}\}.$$

These indices independently refer to some element in either A, B or K. Given that all the elements in B are dependant on those in A, it can be shown that

$$P(A_i) = \sum_h P(A_i|B_h)P(B_h). \tag{2.7.4}$$

By substation of equation (2.7.3), it is shown that

$$P(A_i) = P(A_i) \sum_h \frac{P(B_h|A_i)P(B_h)}{\sum_j P(B_h|A_j)P(A_j)}.$$

The desired probability $P(A_i)$ now appears on both sides. Estimating $P(A_i)$ allows this to be turned into an iterative procedure,

$$P_{r+1}(A_i) = P_r(A_i) \sum_h \frac{P(B_h|A_i)P(B_h)}{\sum_j P(B_h|A_j)P(A_j)}. \tag{2.7.5}$$

By assuming that $P_0(A_i) = c$, $c \in \Re$,

$$P(A_x) = \frac{A_x}{\sum_n A_n}, \quad P(B_y) = \frac{B_y}{\sum_n B_n} = \frac{B_y}{\sum_n A_n} \text{ and } P(K_{x,y}) = \frac{K_{x,y}}{\sum_m K_m}$$

the equation (2.7.5) is simplified to

$$A_{i,r+1} = A_{i,r} \sum_h \frac{K_{h,i} B_h}{\sum_j K_{j,k} A_{j,r}}. \tag{2.7.6}$$

Equation (2.7.6) may then generally be expressed in terms of convolution, that is

$$A_{i,r+1} = A_{i,r} \sum_h \frac{K_{h-i+1} B_h}{\sum_j K_{j-k+1} A_{j,r}}$$

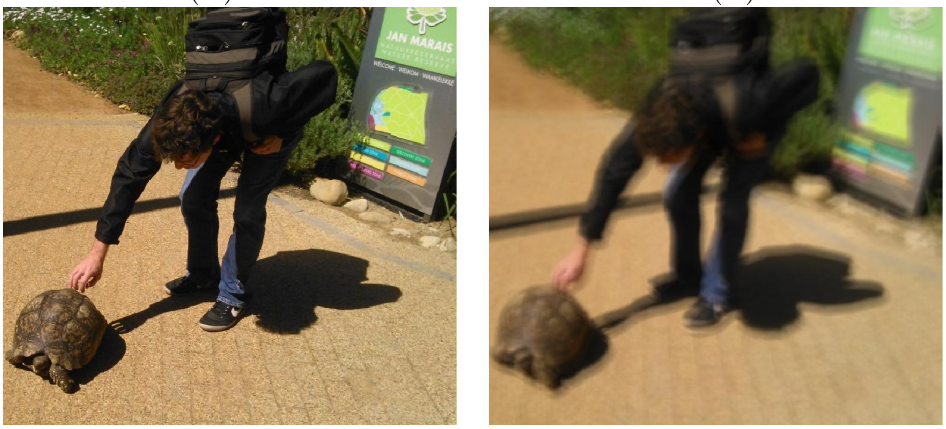
$$\text{or rather } A_{r+1} = A_r \cdot \left[\acute{K} * \frac{B}{A_r * K} \right].$$

Here $\acute{K}$ is the reverse of the kernel that is $\acute{K}_n = K_{m-n}$. This one dimensional result can similarly be expanded to apply for two dimensional deconvolution.

2.7.4 Deconvolution results

Here a test image was blurred with the simulated blur kernel to produce a blurred image, see Figure 2.9. The Weiner and Lucy-Richardson deconvolution methods were then tested, by deconvolving the blurred image with the known blur kernel. Results were produced for varying signal to noise ratios with the Weiner filter, see Figure 2.10, and varying number of iterations for the Lucy-Richardson deconvolution method, see Figure 2.11. Lastly, the best deconvolution results from both methods were compared, see Figure 2.12.

(A) (B)



(K)

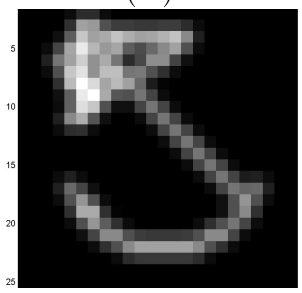
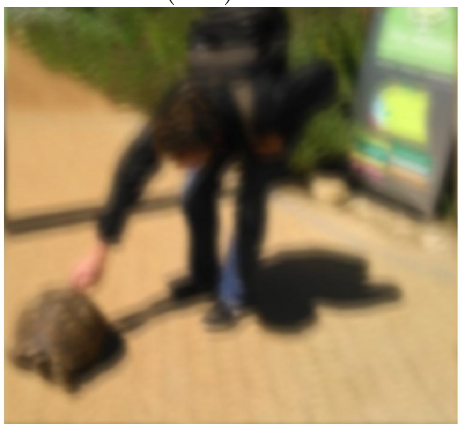


Figure 2.9: The original test image (A), blurred test image (B) and blur kernel (K).

Weiner Filter

(A) (B)

(W1) (W2)



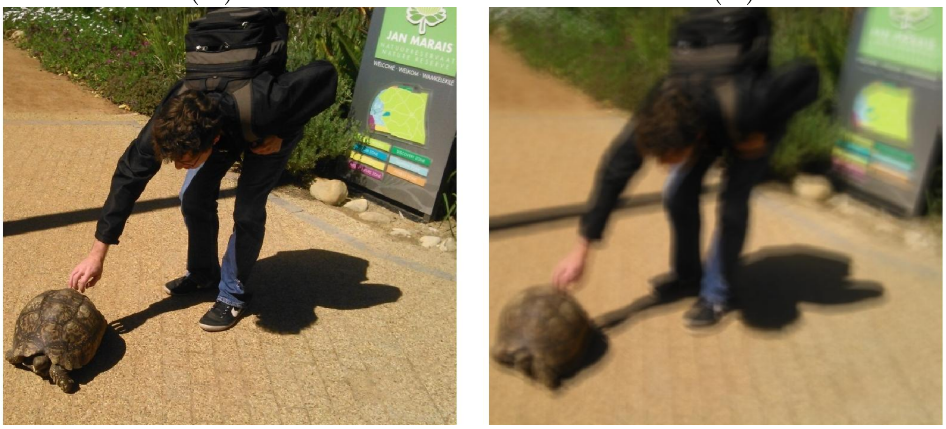
(W3) (W4)

Figure 2.10: The blurred image (B) and original image (A). Weiner filtered results of the blurred test image with signal to noise ratio k equal to (W1):$k = 10$, (W2):$k = 1$, (W3):$k = 0.1$, (W4):$k = 0.01$.

It can be seen from Figure 2.10:(W1), that a high estimate of the SNR yields no deblurring at all. As the SNR estimate gets closer to the actual SNR value the deblurring starts to yields better results.

Richardson Lucy deconvolution

(A) (B)



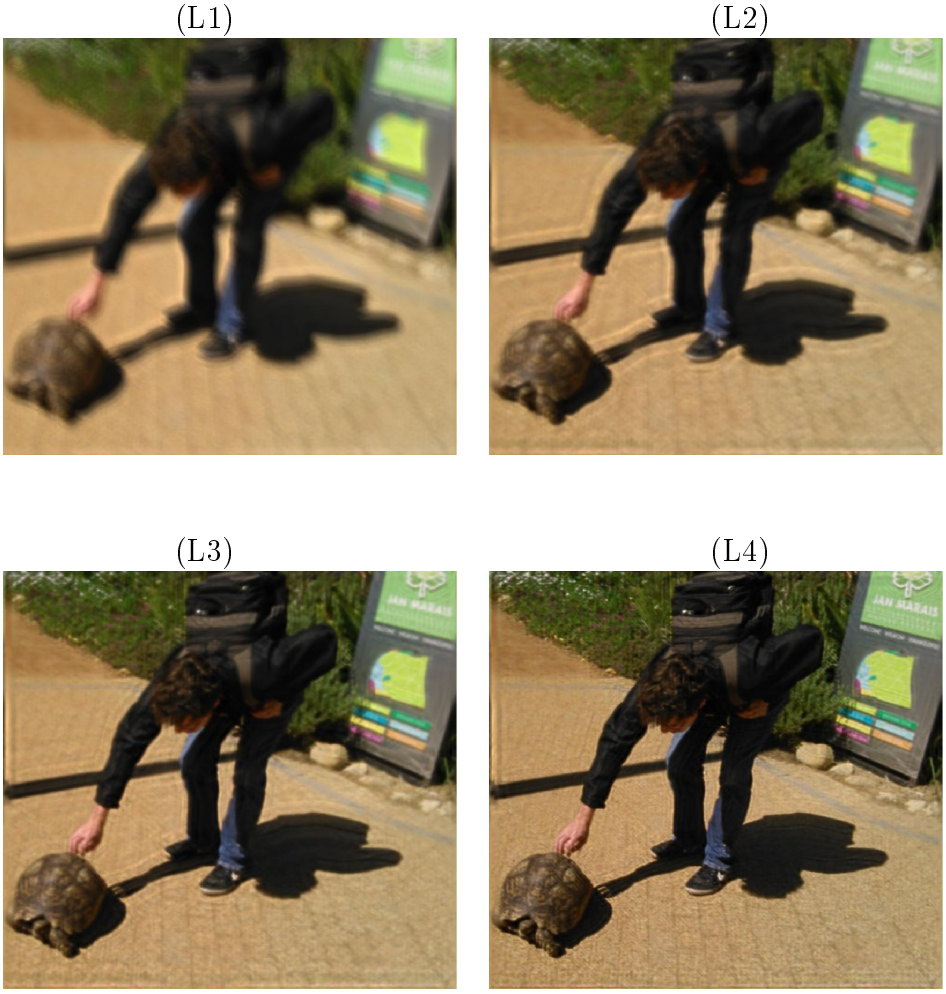


Figure 2.11: Deconvolved using the Lucy-Richardson algorithm. The blurred image (B) and original image (A). Deconvolved results of the blurred test image with number of iterations i equal to (L1):$i = 1$, (L2):$i = 5$, (L3):$i = 10$, (L4):$i = 20$.

As the number of iteration are increased the image becomes notably sharper. However, it can be seen by comparing Figures 2.11:(L3),(L4), that apply to many iterations yields an increase in ringing artifacts.

Weiner Filter Vs Lucy-Richardson deconvolution

Figure 2.12: Comparison of Lucy-Richardson to Weiner filtering. The blurred image (B) and original image (A). Deconvolved results of the blurred test image with, the Weiner filter $k = 0.1$ (W), the Lucy-Richardson deconvolution $i = 10$ (L).

Both Wiener filtering and the Lucy-Richardson deconvolution algorithms display considerable ability to deblur images when the Blur kernel is known. To set a standard, all images that will deblurred throughout the remainder of this thesis, will be deconvolved using the Lucy-Richardson deconvolution for ten iterations.

Chapter 3

Related work

3.1 Literature survey

The deblurring of images and deconvolution is a topic that has been studied for decades. Once the blur kernel is known, implementations of algorithms such as Weiner deconvolution, Weiner (1949), or Lucy-Richardson deconvolution, Lucy (1974); Richardson (1972), may be utilized to restore a blurred image.

Therefore much of the recent work in the area of image deblurring has focused on the estimation of the *blur kernel* [1] itself.
Previous work done on blind image deconvolution makes use of optimization techniques to estimate the blur kernel. Joshi *et al.* (2008) proposes an edge reconstruction technique where blurred edges are sampled to predict previously sharp edges and create a latent sharp image estimation. This latent estimation is used to identify the kernel parameters by minimizing the difference between the known blurred image and a reconstructed blurred image, which is the result of convolution of the estimated sharp image with a predicted blur kernel. This technique is successfully used to identify linear blur but does not perform non-linear kernel estimation. Fegus *et al.* (2006) present an algorithm which is successful in estimating non-linear kernels by minimizing the effect of the blur kernel on the histogram of log image gradients.

The Radon transform has been used to estimate linear kernels that model motion blur along a straight line, Sun *et al.* (2009) and was later improved to include the estimation on non-linear or extreme motion blur, Cho *et al.* (2011). The basis of the algorithm that is presented in this thesis is described by, Cho *et al.* (2011), however we will show that improvement of the basic tomographic estimation algorithm can be done by using sinogram interpolation. We shall now briefly describe the work done by, Joshi *et al.* (2008); Fegus *et al.* (2006) and Cho *et al.* (2011).

[1] often referred to as the image point spread function (PSF).

3.2 Joshi et al.

In Joshi *et al.* (2008), an algorithm is presented by which both blind and non-blind blur kernel estimation can be done. The blur kernel estimation is based upon the following argument:
Given "small" amounts of image blur, the location of image features, in particular edges, can still be detected. Using these features an estimation of the underlying sharp image can be found. Assuming that a detected edge in a blurred image was a step edge before blurring, it can be shown that pairs of predicted sharp and blurred edges provide information about the radial profile of the desired blur kernel.

Therefore, if a blurred image contains enough edges spanning a sufficient number of different orientations, the blurred image and predicted sharp image can be used to solve for a general two-dimensional blur kernel.

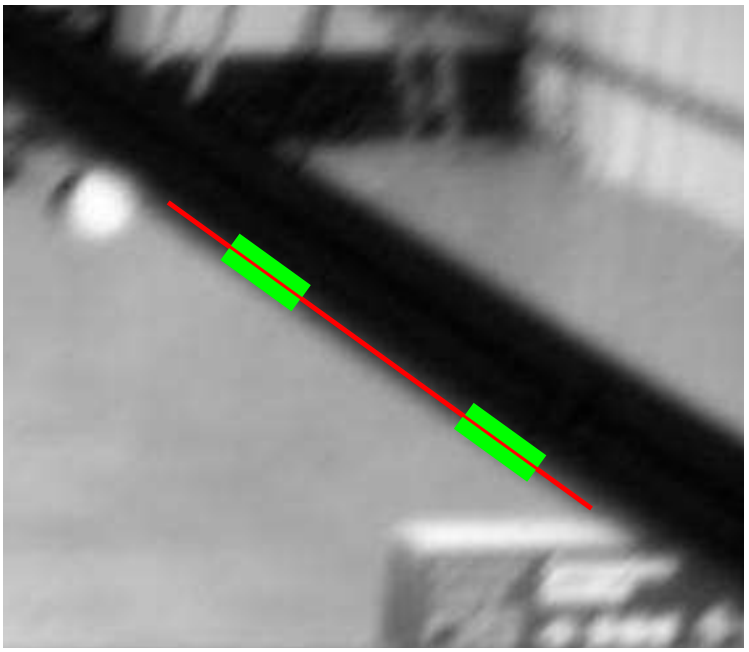


Figure 3.1: An example of edges identified in a blurred image.

Figure 3.1 shows edges that may be identified in the blurred image. The green bands indicate pixels found within the boundary radius[2] while the red marked pixels indicate the predicted previously sharp edge location. We shall now briefly discus the procedure developed by Joshi *et al.* (2008).

[2]All pixels covered by and inside of the green bands will later be used as a masking function to help with the MAP approximation.

3.2.1 Sharp image estimation

In order to estimate the sharp image $A(x, y)$ from the observed blurred image $B(x, y)$, it is assumed that blur is due to a blur kernel with a single mode. Therefore, when an image is blurred, the original position of the edge can still be estimated, however the profile of the edge is changed. The following is a detailed description of how an estimated sharp image $A'(x, y)$ is obtained from the blurred image $B(x, y)$.

Edges in the blurred image are located using a difference of a Gaussians edge detector and the edge orientations are obtained[3]. This allows the extraction of the *edge profile* by sampling along the edge normal.

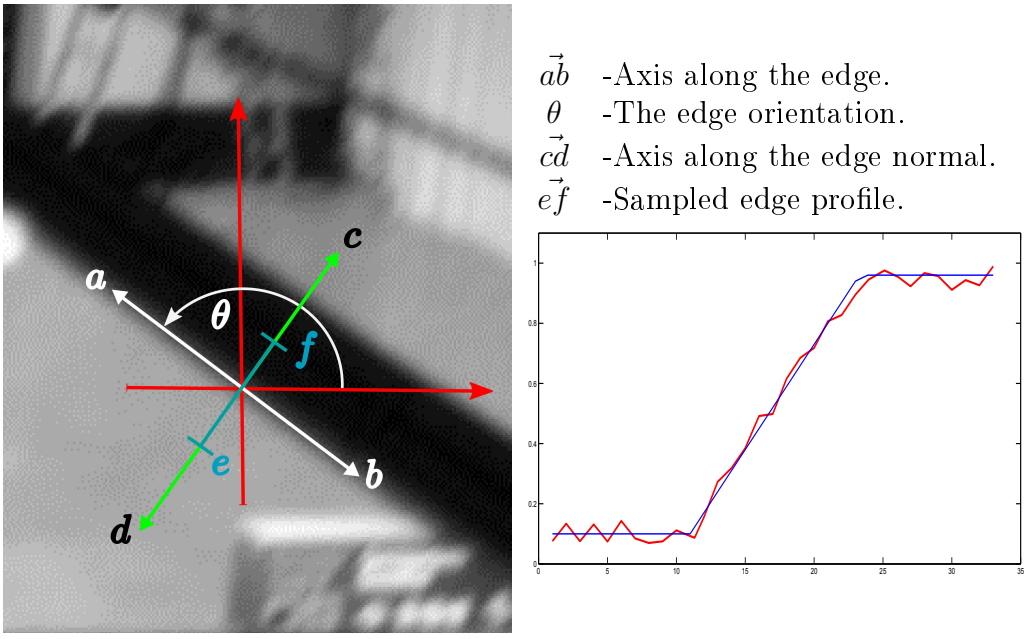


Figure 3.2: Sampling the edge profile.

Figure 3.2.1 shows an edge in the test image at angle θ and the sampled edge profile. An ideal sharp edge can be estimated by finding the local maximum and minimum pixel values along the edge profile. The maximum and minimum values are then "propagated" inward from each side of the edge to a *sharp edge location*, see Figure 3.2.1. However the method by which the authors complete their "propagation" step is not made clear in their paper. Here we will describe a procedure that we believe produces similar results to the unclarified "propagation" procedure.

[3]See chapter 5 for further information.

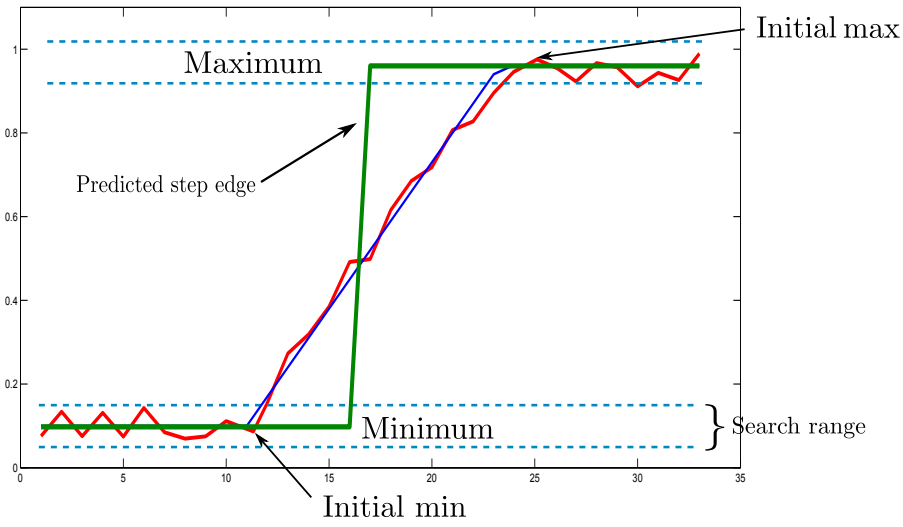


Figure 3.3: Edge profile illustration, moving along the edge profile from left to right to find the minimum and maximum values and produce a predicted step edge.

In order to find the maximum value, the algorithm follows an edge profile $\vec{q}$, sampling the pixel intensity $q_i \in \vec{q}$, looking for a local maximum using hysteresis. Specifically the maximum pixel is the first pixel that is less than or equal to 90% of the value of the previous pixel i.e.

$$\mathrm{Mx} = q_i \leq 0.9 q_{i-1}.$$

Similarly, the minimum is given by

$$\mathrm{Mn} = q_j \geq 0.9 q_{j-1}.$$

Once these initial values and locations are identified, the minimum and maximum are stored as the mean of all the pixel values along the edge profile that are within 10% of the initial maximum or minimum value. The minimum and maximum values are then propagated to a sub-pixel location $a' = (x', y')$ (the estimated location of the original sharp edge). The pixel $A'_{x,y}$ closest to a', will now form part of a predicted sharp edge image. It is given an intensity, which is weighted by the average of the maximum and minimum values according to the distance of the sub-pixel location to the pixel center (x, y), that is

$$A'(x, y) = \frac{(1 - l)\,\mathrm{Mn} + l\,\mathrm{Mx}}{2}$$

Since intensities can only be estimated reliably near the predicted edge locations, only pixels within a certain radius of the predicted sharp values are kept. The reliable pixel positions are saved as an image mask M.
The above procedure provides us with an estimate for the sharp edge image $A' \approx A$.

3.2.2 Blur kernel estimation

Once the sharp image $A'(x, y)$ has been estimated, the blur kernel $K(x, y)$ is found by estimating a kernel, which when convolved with $A'(x, y)$, produces an image close to blurred input image B. This is done using a Bayesian framework and solved using a *maximum a posteriori* (MAP) technique.

The most likely estimate for the blur kernel $K(x, y)$ is given by the maximization of the posterior distribution $P(K \mid B)$, given the estimate $A'(x, y)$, the observed blurred image $B(x, y)$ and the image formation model $B(x, y) = A(x, y) * K(x, y) + N(x, y)$.
Using Bayes' rule, it is shown that

$$P(K \mid B) = P(B \mid K)P(K)/P(B),$$

with

$$P(K \mid B) \propto L_l(K \mid B),$$

where $L_l(K \mid B)$ is the log likelihood of $K(x, y)$ given $B(x, y)$. It is then shown that

$$L_l(K \mid B) = L_l(B \mid K) + L_l(K),$$

and furthermore that

$$\begin{aligned} \text{argmax}_K(P(K \mid B)) &= \text{argmax}_K(L_l(K \mid B)) \\ &= \text{argmax}_K(L_l(B \mid K) + L_l(K)). \end{aligned}$$

Therefore an estimation of the blur kernel can be found minimizing the negative log likelihood of the posterior distribution, given that

$$\text{argmax}_K(P(K \mid B)) = \text{argmin}_K(N_l(B \mid K) + N_l(K)),$$

where $N_l(X)$ is the negative log likelihood of X.

The terms of the negative log likelihood equation are determined as follows: Given the image formulation model, the blurred and the predicted sharp images, the data term ($N_l(B \mid K)$), becomes

$$N_l(B \mid K) = ||M(N) - M(A' * K)||^2/\sigma^2,$$

where $M(A)$ is a *masking function* that only allows the use of pixels within a certain radius of the predicted sharp edge pixels. σ^2 normalizes the data term according to the variance of the noise found on the blurry image by using a technique presented in Liu *et al.* (2006).
The kernel term $N_l(K)$ is then used to regularize and model the prior assumptions that have been made about the blur kernel. Accordingly it is given that

$$N_l(K) = \lambda\gamma||K||^2,$$

where λ controls the smoothness penalty applied to the kernel in order to avoid large gradients. The term

$$\gamma = (2R+1)^2,$$

normalizes the kernel area using an estimate of the kernel radius R. Lastly the negative log likelihood equation is then given by

$$\text{argmax}_K(P(K \mid B)) = \text{argmin}_K(N_l(K \mid B))$$

$$= \text{argmin}_K \left[||M(N) - M(A' \otimes K)||^2/\sigma^2 + \lambda\gamma||K||^2\right]$$

such that $\forall k_{i,j} \in K$, $k_{i,j} \geq 0$.

3.3 Fergus, et al

In Fegus *et al.* (2006), an algorithm is presented for the estimation of non-linear motion blur kernels. We shall briefly discuss the theoretical background developed for a blur kernel estimation procedure, as seen in Fegus *et al.* 2006. This kernel estimation algorithm is based upon the idea that image gradients are reduced when the image is blurred. If a model of the original sharp image gradients can be obtained and the gradient reduction measured, a blur kernel can be estimated.

Research into *natural image* statistics have been shown to be successful in many applications see Miskin and Mackay (2000). Figure 3.5 shows a typical log histogram obtained from the sharp test image. Generally the distribution of gradients has high values for small image gradients (those gradients close to zero) and lower values for high gradients. This confirms what is to be expected, as photographs often contain large regions of almost constant intensity these regions will produce small image gradients. The higher image gradients are produced by regions where there are rapid changes in image intensity such as edges, which occur less frequently in photographs.

The image gradients are successfully modeled by the *log histogram of the gradients* in the x and y direction. This is briefly illustrated in the following example where the image gradients, see Figure 3.5, in the x and y directions are measured before and after blurring (see Figure 3.4).

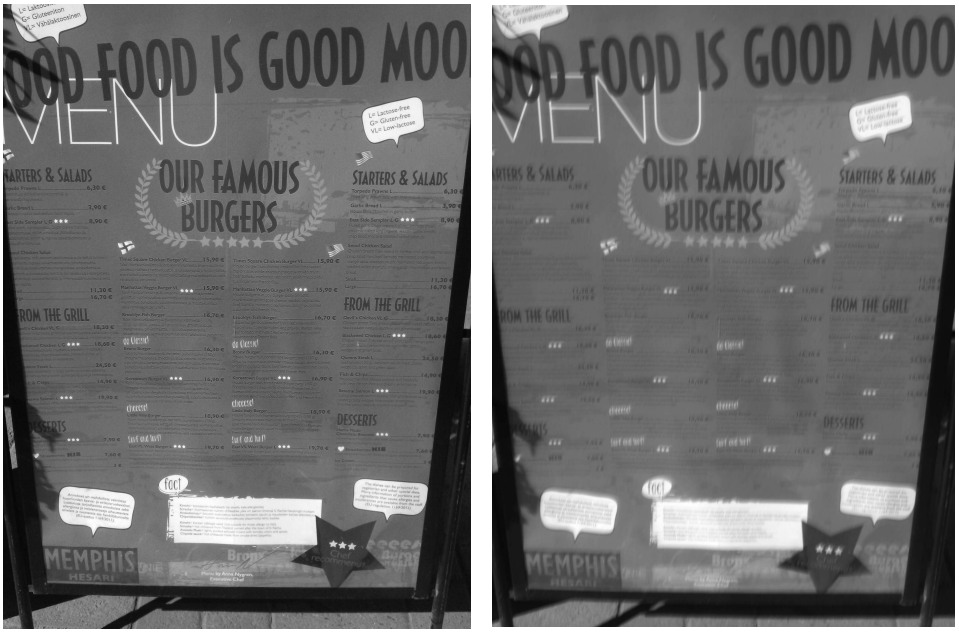


Figure 3.4: Original and simulated blurred image.

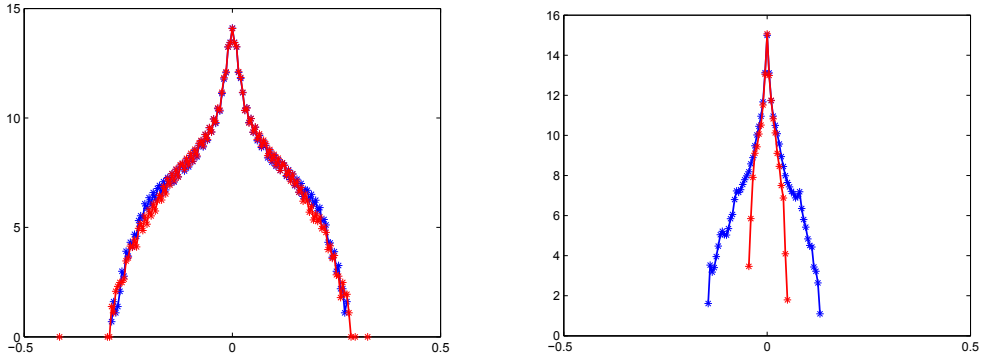
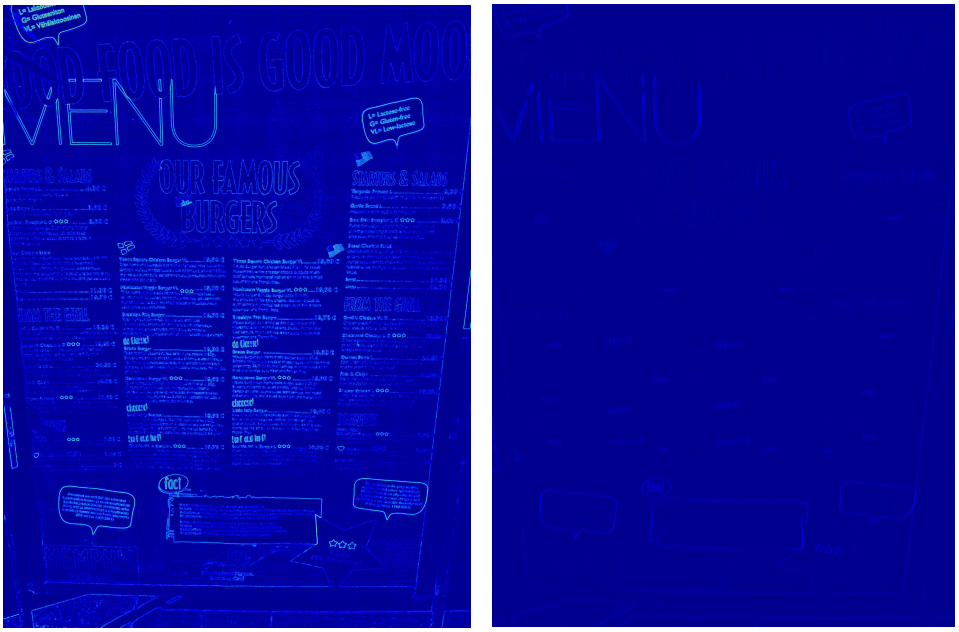


Figure 3.5: Log histograms of the original image gradients and the blurred image gradients. (Gradients along the x-axis in blue, gradients along the y-axis in red.)



(A) (B)

Figure 3.6: Original (A) and blurred (B) image gradients equally scaled. (Light blue to yellow pixels correspond to high image gradients, dark blue regions correspond to low image gradients.)

The two images in Figure 3.6 show the intensity change of the image gradients. The *intensity of the image gradient* for an image A is calculated as follows

$$\|A'\| = \sqrt{\left(\frac{\partial A}{\partial x}\right)^2 + \left(\frac{\partial A}{\partial y}\right)^2} = \sqrt{(\nabla_x A)^2 + (\nabla_y A)^2},$$

where $\nabla_x A$ refers to the gradient of the image A in the x direction. The lack of light blue regions and increase in dark blue in the gradients of the blurred image, Figure 3.6:(B) shows that the intensity of the image gradients are reduced. The reduction effects of the blur are most evident in regions where the image contains a high density of edges.

3.3.1 Blur kernel estimation

In order to estimate the blur kernel from the log histogram of gradients, a model which represents the distribution over gradient magnitudes is modeled by a zero mean Gaussian mixture

$$\sum_{c}^{C} \pi_c \mathrm{N}(0, v_c),$$

with variances v_c and weights π_c .
The model $p(\nabla A)$ forms a basis for prior assumptions about the sharp image gradients. Given a blurred image $B(x, y)$, the kernel $K(x, y)$ and the sharp image $A(x, y)$ are estimated by finding the image and the kernel with highest probability, given the prior $p(\nabla A)$. The optimization is performed in the gradient domain, because image gradients are used as the data and the blurring model still applies in the gradient domain, that is

$$\nabla B(x, y) = \nabla A(x, y) * K(x, y) + N(x, y),$$

since convolution is a linear operation. It is assumed that the noise is Gaussian with variance σ^2.
The prior $p(K)$ ensures that all entries of $K(x, y)$ are positive. Fegus *et al.* (2006) specifically choose the kernel prior to be a mixture of D exponential distributions

$$\sum_{d}^{D} \pi_d \mathrm{E}(\lambda_d),$$

with scale factors λ_d and weights π_d. Given the measured image gradients ∇B, the posterior distribution over the unknowns with Bayes' rule is given by

$$p(K, \nabla A|\nabla B) \propto p(\nabla B|K, \nabla A)p(\nabla A)p(K) \tag{3.3.1}$$

$$= \prod_i \mathrm{N}(\nabla B(i)|(K * \nabla A(i)), \sigma^2) \tag{3.3.2}$$

$$= \prod_i \sum_{c=1}^{C} \pi_c \mathrm{N}(\nabla B(i)|0, v_c) \prod_j \sum_{d=1}^{D} \pi_d \mathrm{E}(K_j|\lambda_d) \tag{3.3.3}$$

where i indexes over image pixels and j indexes over blur kernel elements. It is assumed that the gradients in ∇B are independent, as well as the elements in ∇A and K. A MAP search is used to obtain the solution that maximizes $p(K, \nabla A|\nabla B)$.

3.4 Cho, et al

Cho *et al.* (2011) describe an advanced version of the tomographic transform kernel estimation technique, first described by Sun *et al.* (2009), with which more intricate motion blur kernels can be estimated. The basis of the algorithm that is described in Cho *et al.* (2011), which will be referred to as *the direct inversion method*, has the same basis as the overall algorithm covered in this thesis.
As the focus of this thesis was to study the tomographic transform estimation technique this overview will not cover all the theory directly, rather an outline of the presented algorithm and mathematics is described.

It will later be shown that radial information about the image blur kernel can be obtained by detecting and sampling edge profiles with in the blurred image. Once these edge profiles have been captured they can be use to construct a sinogram of the blur kernel. However the obtained sinograms are usually sparse and random since images are not guaranteed to have edges oriented in any directions. It will later also be shown that the sparsity of the sinogram causes reconstruction artifacts when it is inverted using the inverse Radon transform.

It is here that the algorithm that we will present and the algorithm of Cho *et al.* (2011) differ. In Cho *et al.* (2011) a Bayesian formulation is used to incorporate assumptions of sparsity and smoothness about the blur kernel. By reformulation of the Bayesian framework into likelihood terms where

$$p(K|B) \propto p(B|K)p(K),$$

the posterior probability $p(K|B)$ of the blur kernel K given the observed blurred image B is modeled as

$$p(B|K) \propto exp\left(-\frac{1}{2\eta_{obs}^2}\sum_{i=1}^{N}||S(i) - R(K,\theta_i)||^2\right).$$

Where $S(i)$ is the obtained sampled sinogram column from the blurry image and $R(K,\theta_i)$ the Radon transform of the estimated kernel at angle θ_i. Here η_{obs}^2 is the variance of observation noise that comes from the image. The kernel prior is modeled as

$$p(K) \propto exp\left(-\lambda_1||K||^{\gamma_1} - -\lambda_1||\nabla K||^{\gamma_2}\right),$$

here the contestants $\lambda_1, \lambda_2, \gamma_1, \gamma_2$ help to incorporate the assumed knowledge that the profiles of blur kernels and their gradient profiles are sparse.
These priors are then used to solve for the kernel by minimising the negative log likelihood of the posterior probability, that is,

$$-\log(p(B|K)p(K)).$$

The final result is what Cho *et al.* (2011) refer to as the RadonMAP algorithm.

Chapter 4

The tomographic reconstruction algorithm

4.1 The Radon transform

Definition 4.1. The *Radon transform*, Helgason (1999); Toft (1996), of an image $A(x, y)$, is defined as

$$\mathcal{R}(\theta, r) = \mathfrak{R}[A(x, y)] = \int_{-\infty}^{\infty} A(r\cos\theta - t\sin\theta, r\cos\theta + t\sin\theta)dt \qquad (4.1.1)$$

with

$$(x, y) = (r\cos\theta - t\sin\theta, r\cos\theta + \sin\theta). \qquad (4.1.2)$$

By calculating the Radon transform of an image $A(x, y)$, for a set angles $\theta_k \in \Theta$, a *sinogram* of the image is obtained. The sinogram of an image $A(x, y)$ is the collection of column vectors

$$\mathcal{R}(\theta, r) = [\vec{s}_{\theta_1}(r), \ldots, \vec{s}_{\theta_k}(r)], \text{ where } \vec{s}_{\theta_k}(r) = R(\theta_k, r),$$

produced by implementation of the forward Radon transform, such that $\theta_k \in [0, 2\pi]$ and $k \geq 1$.

Crucially it can be shown that a sinogram can be inverted to recover the original image. Mathematically the original image can be reconstructed using the central slice theorem Gonzalez and Woods (2008): page 396. To assure our selves of this we briefly investigate a proof of the central slice theorem[1].

[1] A sinogram of an image can then be used to reconstruct the image using what is known as filtered back projection Kak and Slaney (2001).

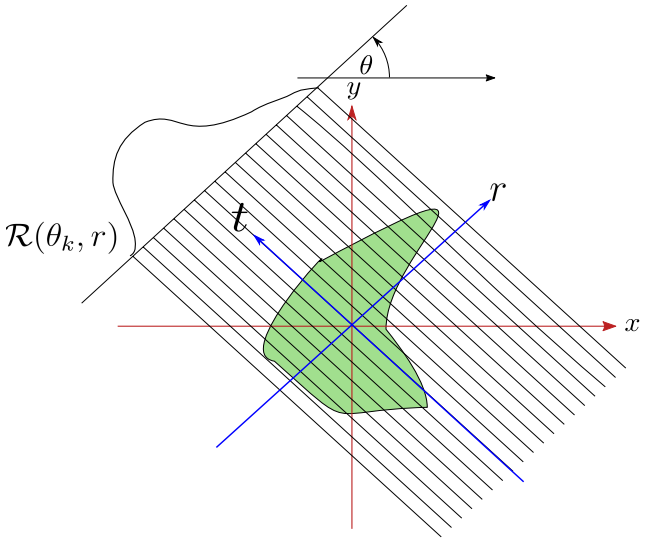


Figure 4.1: The Radon transform.

The Radon transform $\mathcal{R}(\theta_k, r)$ of an image $A(x, y)$ onto the r axis, at angle θ_k is given by,

$$R(\theta_k, r) = \int_{-\infty}^{\infty} A(r\cos\theta - t\sin\theta, r\cos\theta + t\sin\theta)dt.$$

Then the Fourier transform of $A(x, y)$ is

$$\hat{A}(u, v) = \mathfrak{F}[A(x, y)] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} A(x, y)e^{-2\pi i(ux+vy)}dx\,dy.$$

Setting $v = 0$ the Fourier transform of $A(x, y)$ gives,

$$\hat{A}(u, 0) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} A(x, y)e^{-2\pi i(ux)}dx\,dy.$$

Substituting $(x, y) = (r\cos\theta - t\sin\theta, r\cos\theta + \sin\theta)$ and reversing the order of integration it can be shown that

$$\hat{A}(u, 0) = \int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty} A(r\cos\theta - t\sin\theta, r\cos\theta + \sin\theta)dt\right] e^{-2\pi i(u(r\cos\theta - t\sin\theta))}dr,$$

$$\mathfrak{F}[\mathcal{R}(\theta_k, r)]_{r\to u} = \int_{-\infty}^{\infty} \mathcal{R}(\theta_k, r)e^{-2\pi i(u(r\cos\theta - t\sin\theta))}dr.$$

This is the Fourier transform of the Radon transform $\mathcal{R}(\theta_k, r)$ at θ_k.

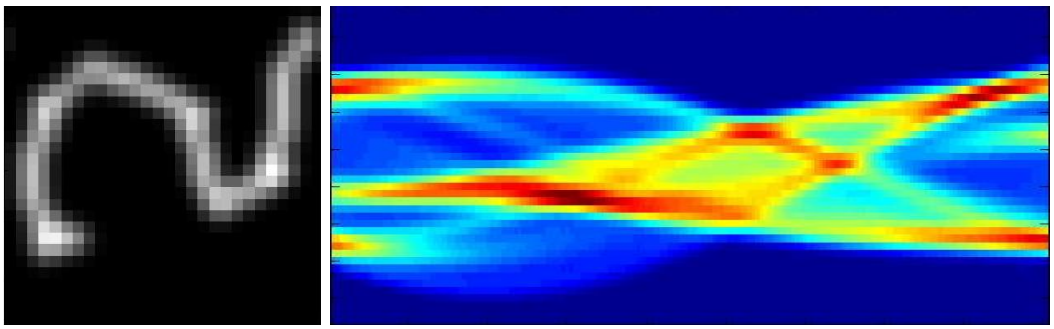


Figure 4.2: A blur kernel and kernel sinogram.

4.2 Kernel estimation using Radon transform

A continuous model of a blurred image $B(x, y)$ with no added noise, is given by

$$B(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} A(x',y')K(x-x',y-y')dx'\,dy'. \tag{4.2.1}$$

Here $A(x, y)$ is the scene image and K is the unknown blur kernel that is to be estimated. Equation (4.2.1) may be expressed along any axes by applying the transformation

$$\begin{bmatrix} r \\ t \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}. \tag{4.2.2}$$

This rotates the reference axes about the point (x, y), counterclockwise by some angle θ. The image A in (r, t) coordinates at angel θ will be denoted by

$$A_\theta(r,t) = A(r\cos(\theta) - t\sin(\theta), t\cos(\theta) + r\sin(\theta)).$$

Using this equation (4.2.1) is transformed, to

$$B_\theta(r,t) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} A_\theta(r',t')K_\theta(r-r',t-t')dr'\,dt'. \tag{4.2.3}$$

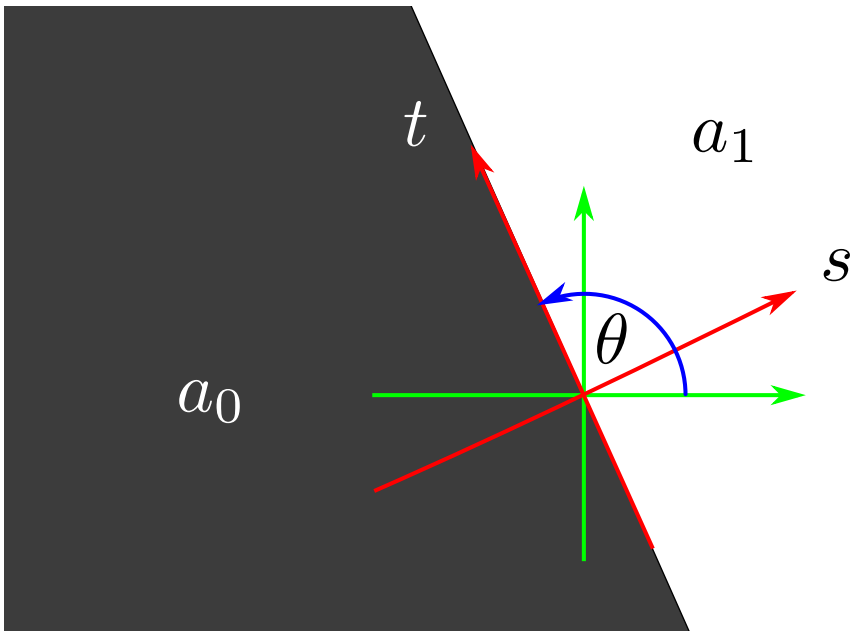


Figure 4.3: Illustration of blurred ideal edges.

Consider a sharp edge in a scene image A_θ, where

$$A_\theta = \begin{cases} a_0, & r < 0 \\ a_1, & r > 0 \end{cases}.$$

When $t = 0$, it is found that

$$B_\theta(r,0) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} A(r',t')K_\theta(r-r',-t')dr'\,dt'. \tag{4.2.4}$$

$$= \int_{-\infty}^{\infty}\int_{r=-\infty}^{r=0} A(r',t')K_\theta(r-r',-t')dr'\,dt' + \int_{-\infty}^{\infty}\int_{r=0}^{r=\infty} A(r',t')K_\theta(r-r',-t')dr'\,dt'$$

$$= \int_{-\infty}^{\infty}\int_{r=-\infty}^{r=0} a_0K_\theta(r-r',-t')dr'\,dt' + \int_{-\infty}^{\infty}\int_{r=0}^{r=\infty} a_1K_\theta(r-r',-t')dr'\,dt'$$

Applying the substitution $u = r - r'$ with $du = -dr'$, gives the equation

$$B_\theta(r,0) = a_0\int_{-\infty}^{\infty}\int_{u=r}^{u=\infty} K_\theta(u,-t')du\,dt' + a_1\int_{-\infty}^{\infty}\int_{u=-\infty}^{u=r} K(u,-t')du\,dt'. \tag{4.2.5}$$

Under the assumption that the blur kernel is energy conserving, that is

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} K(u,v)dudv = 1,$$

it can be shown that

$$B_\theta(r,0) = (a_0 - a_1)\int_{-\infty}^{\infty}\int_{u=\infty}^{u=r} K_\theta(u,-t')du\,dt' + a_0. \tag{4.2.6}$$

The integral of $K_\theta(u, -t')$ can then be expressed as

$$k(\theta, r) = \int_{-\infty}^{\infty} \int_{u=\infty}^{u=r} K_\theta(u, -t')du\, dt' = \frac{B_\theta(r, 0) - a_0}{(a_0 - a_1)}. \tag{4.2.7}$$

By taking the partial derivative of equation (4.2.7) with respect to r, it can be shown that,

$$\frac{\partial k(\theta, s)}{\partial r} = \frac{\partial}{\partial r} \int_{-\infty}^{\infty} \int_{u=\infty}^{u=r} K_\theta(u, -t')du\, dt' = \frac{\partial}{\partial r} \frac{B_\theta(r, 0) - a_0}{(a_0 - a_1)}. \tag{4.2.8}$$

$$\frac{\partial k(\theta, r)}{\partial r} = \int_{-\infty}^{\infty} K_\theta(r, -t')dt'. \tag{4.2.9}$$

With the given transformation equation (4.2.2), this may be written as

$$\frac{\partial k(\theta, r)}{\partial r} = \int_{-\infty}^{\infty} K_\theta(r\cos(\theta) - t\sin(\theta), t\cos(\theta) + r\sin(\theta))dt. \tag{4.2.10}$$

Note that equation (4.2.10) describes the tomographic profile of the kernel as a function of r at some angle θ. This indicates that an estimate for the blur kernel K can be obtained, by applying the inverse Radon transform after having sampled sufficiently many edges at various angles.

4.3 Proposed tomographic algorithm

The theory explained in the previous section provides the working basis for a blur kernel estimation algorithm. The proposed algorithm is laid out as follows:

1. `Find blurred edges and their direction in the blurred image` B.
2. `Sample these edges perpendicular to edge orientation, to produce edge profiles.`
3. `Scale the edge profile according to (4.2.7).`
4. `Take the discrete derivative of the sampled edge derivatives.`
5. `Construct a matrix by using the profile derivatives as columns. Arrange the columns according to the magnitude of the edge orientation in ascending order. This produces an estimated sinogram of the blur kernel.`
6. `Apply the inverse Radon transform to reconstruct the blur kernel from the estimated sinogram.`

The estimated blur kernel can then be used to with known deconvolution methods to produce a deblurred image.

Chapter 5

Edge detection, sampling and selection

5.1 Edge detection

Edge detection plays an important role in many image processing applications. It is also an essential part of blur kernel estimation, since the edges in an image carry radial information about the blur kernel. Edge detection involves finding points in an image where there are sharp changes in image intensity. We are fortunate as this is a field that has been thoroughly studied. However, with the addition of blur, that tends to stretch out the edges or create multiple edges, finding the edge locations can be a bit more of a challenging task.

It will be shown that step functions are typically transformed into gradient functions, when the image is blurred by either linear motion or aperture blur. The appearance of multiple duplicate edges commonly occurs when the blur kernel has an intricate motion path, which has more than one mode in a particular direction.

5.1.1 Blurred edges

As will be explained, there are several ways in which blur can change step edges. These changes hold certain implications for edge detection algorithms and therefore need to be taken into account. Figure 5.2 contains a table with three examples of the aforementioned problems, as caused by various blur kernels.

Figure 5.1: The original test image and its resulting edge image using Canny edge detection.

As can be seen from the first row of the table, the conventional[1] edge detection does not necessarily have a problem in finding blurred edges when the blur is small. However when the image experiences blur due to a longer or more intricate motion path, the edge detection fails (see rows 2 and 3). These problems clearly prompt us to pursue the development of a robust edge detection technique. For this purpose we shall investigate well known edge detection methods such as Canny edge detection.

[1] In this case Canny edge detection as provided by the MATLAB image processing toolbox.

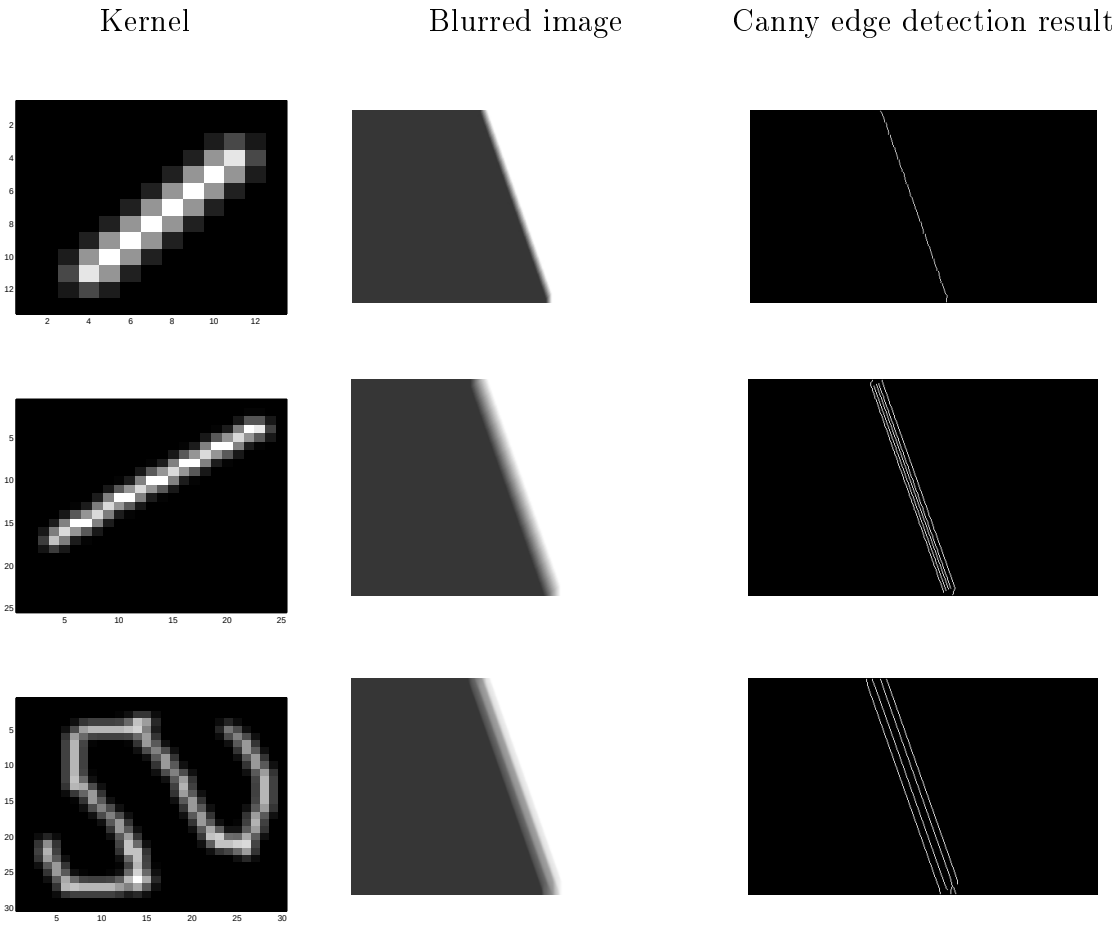


Figure 5.2: Blurred edge images and their respective Canny edge detection results.

5.2 Simple edge detectors

Edge detection can be done in various ways, with gradient methods being favorable for most problems. However simple methods such as the Sobel, Prewitt and Roberts filters, which are a staple of many edge detection algorithms, are not suited to tackle the blurred edge problem. These are basic gradient filters, but they remain useful since they can be used to easily identify the edge

direction. Listed below are various edge detection filters:

$$\text{Sobel:} \qquad F_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad F_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

$$\text{Prewitt:} \qquad F_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, \quad F_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix},$$

$$\text{Roberts cross:} \quad F_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \qquad F_y = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

The edge magnitude and orientation images are obtained by the convolution of the image A and edge detection filter F, they are respectively given by

$$G = \sqrt{G_x^2 + G_y^2}, \quad \Theta = \arctan(G_y, G_x).$$
$$\text{where, } G_x = A * F_x \text{ and } G_y = A * F_y.$$

However, as the focus of this thesis is not edge detection, the discussion on these filters will be limited and the reader may refer to Gonzalez and Woods (2008); Davis (1975); Jahne *et al.* (1999) for further information.

5.3 Advanced edge detection

5.3.1 Canny edge detection

Canny edge detection is a multi-stage algorithm that can be used to detect a wide range of edges. This is a well developed algorithm and John F. Canny produced a computational theory explaining why the technique works. As such, it is a good benchmark to use, when edge detection is discussed. Canny edge detection can be broken down into four steps

- `Reduce the image noise:`
  By applying a 5×5 Gaussian filter, the image is smoothed and noise is reduced.

- `Find the image gradients:`
  This is done by applying the Sobel filters to the image and calculating the edge magnitude G and edge direction Θ.

- `Apply non-maximum suppression to remove false edges:`
  To do this, check if each pixel is a local maximum in its neighborhood in the direction of gradient.

- `Apply hysteresis using an upper and lower threshold:`
  If a pixel gradient is higher than the upper threshold, the pixel is accepted as an edge.
  If a pixel gradient is below the lower threshold, then it is rejected.
  If the pixel gradient is between the two thresholds, then it will be accepted only if it is connected to a pixel that is above the upper threshold.

5.3.2 Partial derivatives of a Gaussian (PDoG)

Canny edge detection provides a good starting point for edge detection, but struggles to identify blurred edges. The derivative of a function can easily be used for edge detection. Derivatives are well suited to highlight edges in the image, because they pick up rapid changes in intensity. Using the *partial derivatives of the Gaussian function (PDoG)* is a common method used for edge detection, as the Gaussian will reduce the effects of image noise. The PDoG images of an image A created using a Gaussian function $g(x, y)$ with $\sigma = 1$ are given by

$$\text{PDoG}_x(x, y) = A(x, y) * \frac{\partial g(x, y)}{\partial x} = A * \frac{x}{2\pi} e^{(x^2+y^2)/2},$$

$$\text{PDoG}_y(x, y) = A(x, y) * \frac{\partial g(x, y)}{\partial y} = A * \frac{y}{2\pi} e^{(x^2+y^2)/2}.$$

5.3.3 Laplacian of Gaussian (LoG) edge detection

The *Laplacian*, Arfken (1985); Stewart (2012), as a variant of the derivative function, also finds image regions where there is a sharp change in intensity. It is therefore often used for edge detection. However since the Laplacian filters are calculating a second derivative of the image, they are more sensitive to noise. It is therefore applied to an image that has first been smoothed with Gaussian filter in order to reduce its sensitivity to noise, and hence the two are often used together.
Noise reduction can be completed as a pre-processing step. However since convolution is associative, this allows the creation of *the Laplacian of Gaussian (LoG)* filter. Combining the Laplacian and Gaussian filters has the benefit of requiring only a single convolution to be performed on the image. The Laplacian operator is defined as

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}.$$

The LoG image of A is given by

$$\text{LoG}(x, y) = A(x, y) * \left[\frac{-1}{\pi\sigma^4} \left[1 - \frac{x^2 + y^2}{2\sigma^2} \right] \exp\left(-\frac{x^2 + y^2}{2\sigma^2} \right) \right]. \tag{5.3.1}$$

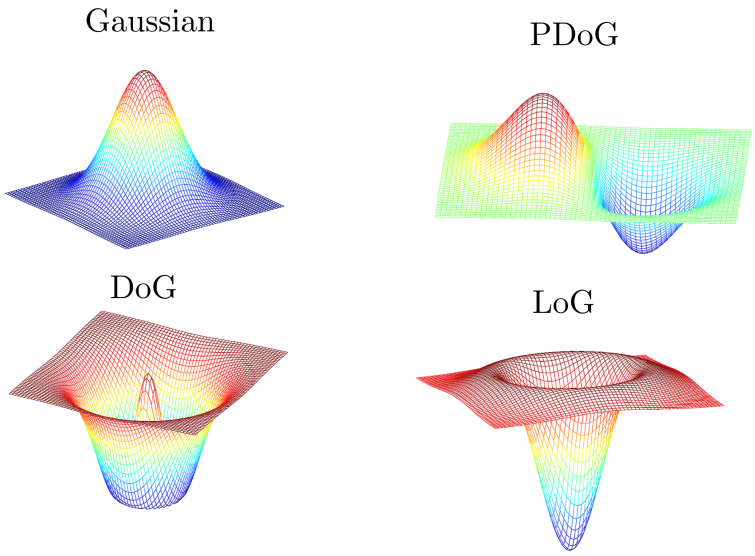


Figure 5.3: A Gaussian function and its related edge detection filters.

5.3.4 Difference of Gaussian (DoG) edge detection

Further edge detection can be done using *the difference of Gaussians (DoG)*. This simply involves subtracting two circular Gaussian functions from each other, where the first and second functions have differing values for sigma. The DoG image of A is given by

$$\mathrm{DoG}(x,y) = A(x,y) * \left[\frac{1}{2\pi}\left[\frac{1}{\sigma_1}\exp\left(-\frac{x^2+y^2}{2\sigma_1^2}\right) - \frac{1}{\sigma_2}\exp\left(-\frac{x^2+y^2}{2\sigma_2^2}\right)\right]\right] \tag{5.3.2}$$

The difference of Gaussians algorithm removes high frequency detail that often includes random noise, and it is therefore suitable for processing images with a high degree of noise.

5.3.5 Dealing with blurred edges

When attempting to locate blurred edges, the edge detection algorithm must identify edges that have been transformed from being one or two pixels thick (typical sharp edges) to possibly tens of pixels in thickness. The elementary edge detection filters, such as the Sobel filter, only measure the intensity change in the image over a width of 3 pixels.

The PDoG, LoG, and DoG functions can all be produced in various filter sizes. This makes them ideal to find wider blurred edges, since the blur kernels themselves may be varying in size. When trying to find blurred edges, it is essential to measure the intensity change over a large enough region, which encompasses the whole of the blurred edge.

5.4 Edge orientation

Derivative filters can be used to obtain the orientation of a edge. It will be shown that the direction orthogonal to the edge is given by

$$\Theta = \arctan(G_y, G_x),$$

where G_x and G_y are the image gradients, found using one of the many available gradient filters. Suppose we have an edge image

$$A(x,y) = \begin{cases} a_1, & x\cos\theta - y\sin\theta \leq 0 \\ a_0, & x\cos\theta - y\sin\theta > 0 \end{cases},$$

rotated to an angle θ from the horizontal. The partial derivatives of $A(x,y)$ with respect to x and y at $x\cos\theta - y\sin\theta = 0$ are given by,

$$\frac{\partial A(x\cos\theta, y\sin\theta)}{\partial x} = A_x(x\cos\theta, y\sin\theta)\sin\theta$$
$$\frac{\partial A(x\cos\theta, y\sin\theta)}{\partial y} = -A_y(x\cos\theta, y\sin\theta)\cos\theta.$$

for which

$$\theta = \arctan(\tfrac{\partial A}{\partial y}/\tfrac{\partial A}{\partial x}) = \arctan\left(\frac{\sin\theta}{\cos\theta}\right).$$

5.5 Detection results

Results obtained by simulating motion blur and then completing edge detection procedure on the blurred image are shown here. Edge detection was then done on the blurred image. Results are shown for Canny edge detection in comparison to that produced by the PDoG and LoG edge detection filters. The original image seen in fig 5.4 was blurred by an kernel that has one mode in the y-direction and a double mode in the x-direction.

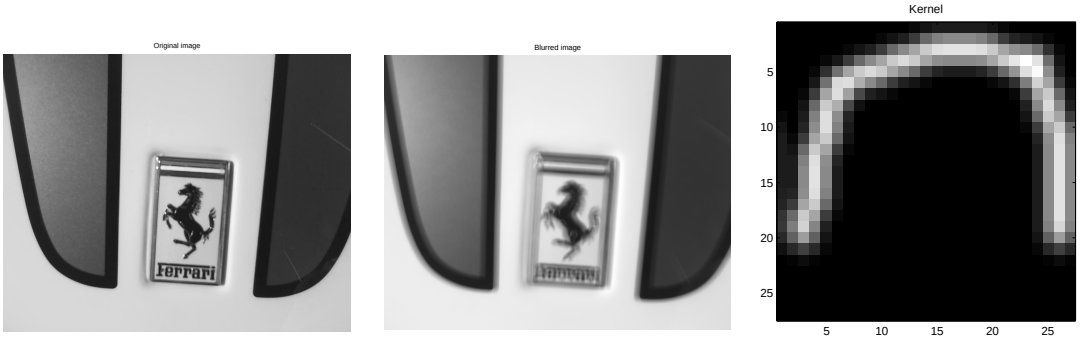


Figure 5.4: The original, blurred image and ground truth kernel.

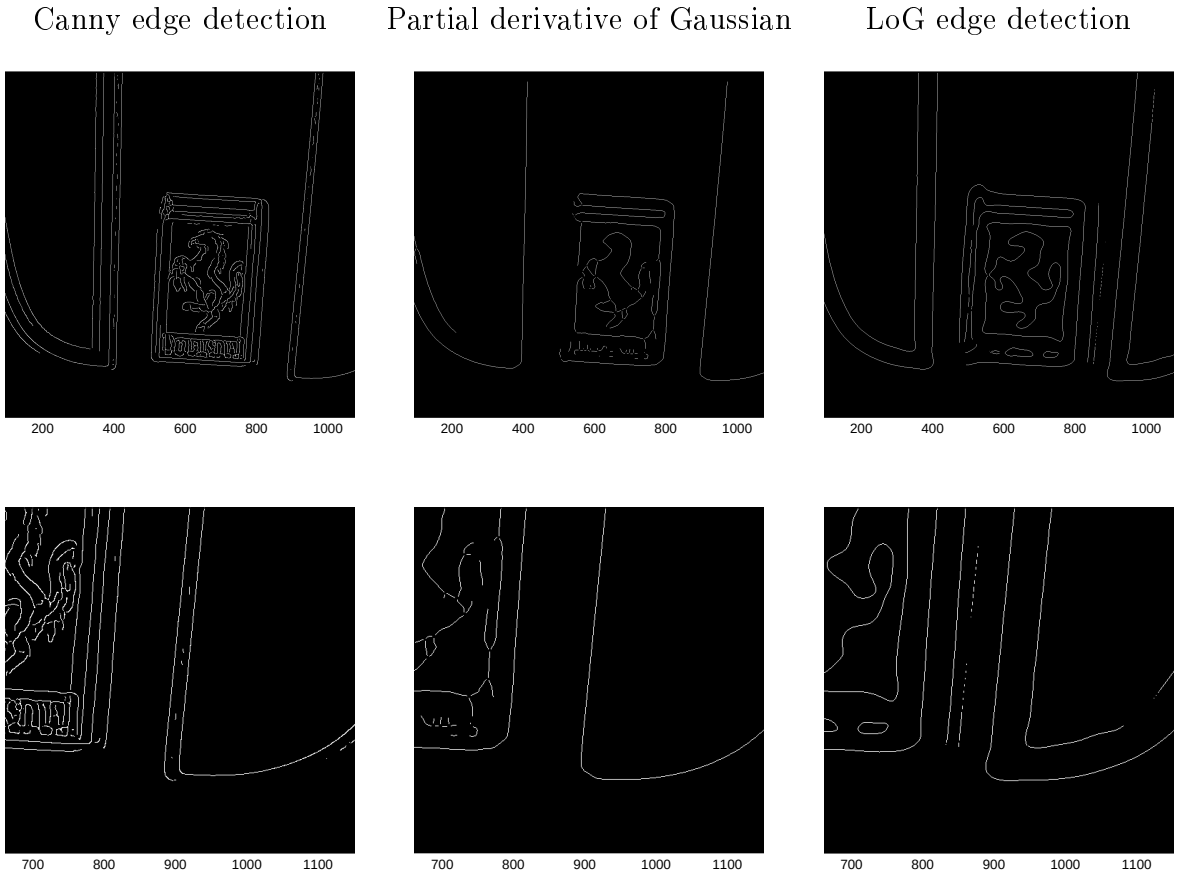


Figure 5.5: Edge detection results.

The example in Figure 5.5 clearly shows the duplicate edges found by Canny edge detection. These duplicate edges are not found when using the PDoG or Log edge detection filters. The results from the latter two are therefore better suited to be used when finding edges in blurred images. We will, however, also require the edge orientation. The PDoG filter is therefore our choice of edge detection, since the PDoG filters G_x and G_y can also be used to find the edge orientation. This gives the partial derivatives of a Gaussian an advantage over the LoG filter, which is a single filter that only finds the edge locations.

5.6 Image sampling

Various methods that can be implemented for image sampling will be discussed here.
For the purposes of this thesis, we will generally be sampling at coordinates

along a straight line. Consider a line segment at angle θ,

$$\vec{c}(t) = [x - t\cos(\theta), y - t\sin(\theta)], \quad t = [-l, l].$$

The mid point of the line will always be located at an integer coordinate. If we let

$$t_m = md \quad \text{with} \quad m = -n, \ldots, n,$$

where d will be called the *sampling distance*, the line is split into $2n+1$ equally spaced sampling coordinates. In this chapter $A_{i,j}$ will refer to the grey scale value of the image at pixel (i, j).

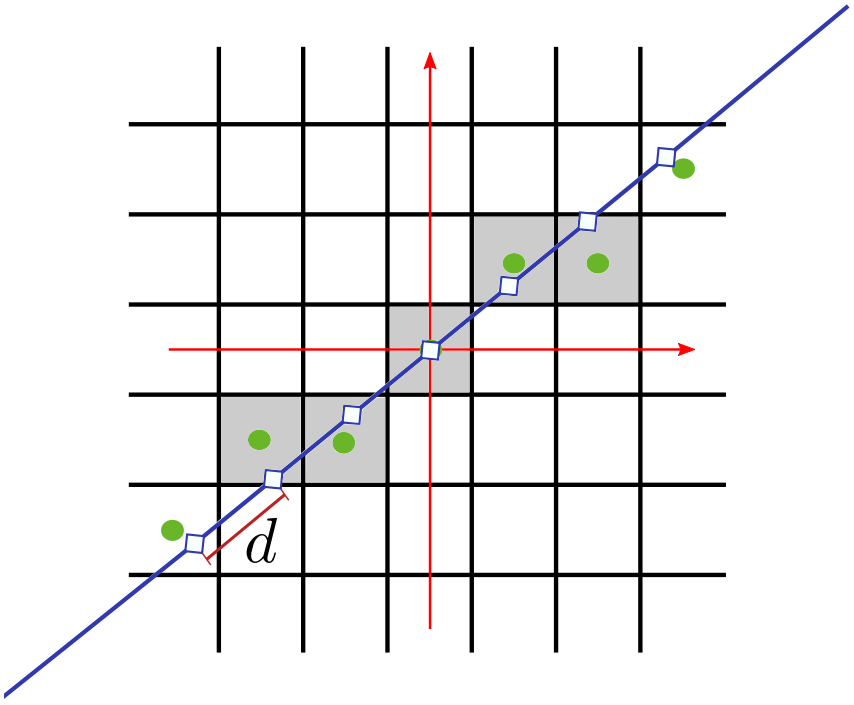


Figure 5.6: Illustration of the sampling line and distance d.

The techniques explained hereafter are applied to edge sampling. It may be helpful to note that these techniques are often used as 2 dimensional interpolation techniques.

5.6.1 Nearest neighbour sampling

The simplest method of image sampling is *nearest neighbour sampling*. In order to sample at a point $\vec{c} = [c_x, c_y]$, the two coordinates are simply rounded to the nearest integers, i.e. the sample s is given by

$$s = A_{\langle c_x \rangle, \langle c_y \rangle},$$

where the rounding function $\langle \cdot \rangle$ is defined as

$$\langle x \rangle = \begin{cases} \lceil x \rceil, & x - \lfloor x \rfloor \geq \frac{1}{2} \\ \lfloor x \rfloor, & x - \lfloor x \rfloor < \frac{1}{2} \end{cases}.$$

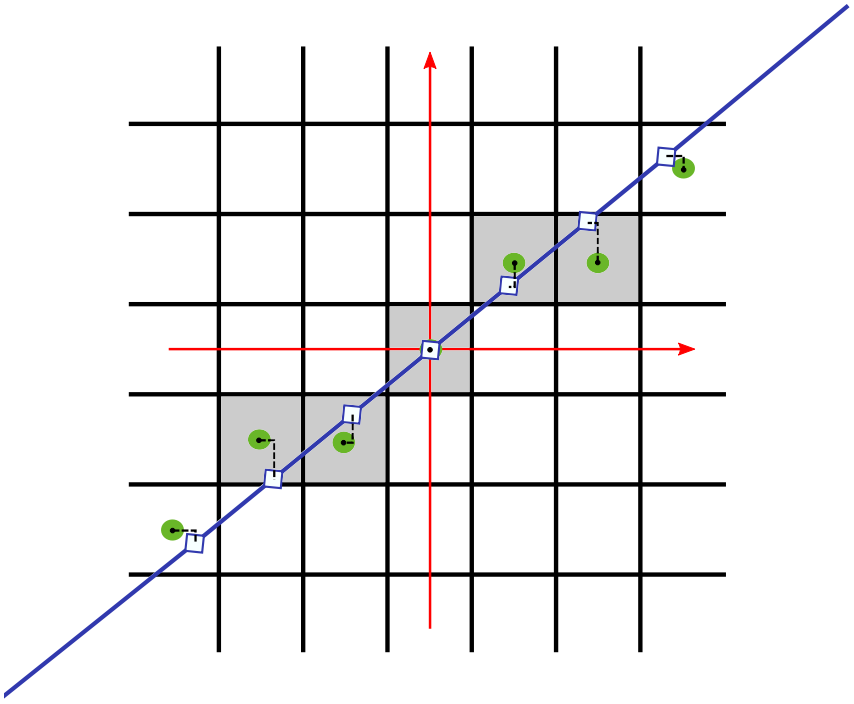


Figure 5.7: Nearest neighbour sampling.

5.6.2 Bilinear sampling

Bilinear sampling is an improvement upon nearest neighbour sampling, in the sense that the obtained sample result is typically smoother, at the expense of some additional computation. To begin, we first need to find the grey scale values of the four pixels that lie closest to the sample position $\vec{c} = [c_x, c_y]$. Let

$$x_f = \lfloor c_x \rfloor, \;\; y_f = \lfloor c_y \rfloor, \;\; y_c = \lceil c_y \rceil, \;\; x_c = \lceil c_x \rceil.$$

By combining the grey scale values of the four pixels that surround $[c_x, c_y]$ as a weighted average, a value for s is obtained. The pixel positions and their distance from $[c_x, c_y]$ are used to calculate the weights for s as follows:

$$w_1^x = x - x_f, \;\; w_2^x = 1 - w_1^x = x_c - x,$$

$$w_1^y = y - y_f, \;\; w_2^y = 1 - w_1^y = y_c - y,$$

where

$$w_1^x, w_2^x, w_1^y, w_2^y \in [0, 1].$$

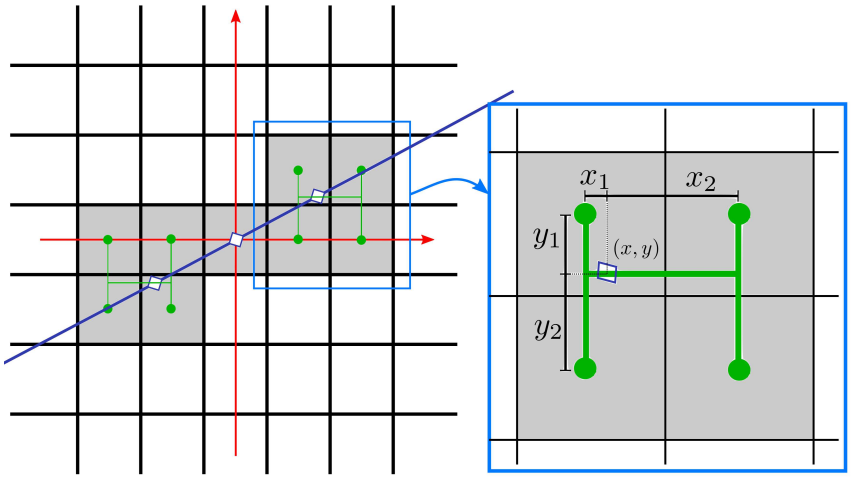


Figure 5.8: Bilinear sampling.

These calculated weights and corresponding grey scale values are combined, such that the sample s is then given by

$$s = w_2^x \left[w_2^y A_{x_f,y_f} + w_1^y A_{x_f,y_c}\right] + w_1^x \left[w_2^y A_{x_c,y_f} + w_1^y A_{x_c,y_c}\right].$$

This allows pixels closer to $[c_x, c_y]$ to be weighted more heavily. Since this scheme is essentially linear interpolation in both the x- and y-directions, it is called bilinear sampling.

5.6.3 Area percentile

A more refined technique can be devised to produce samples with greater accuracy however, this yet again comes at increased computational cost. Bilinear sampling is improvement upon by utilizing the area of the region around the sample point.
This is done by placing a rectangular box around the sample point $\vec{c} = [c_x, c_y]$. The box has a size $2r$ by d and is called a *sample box*, see Figure 5.9. Here r specifies the radius orthogonal to the sample line, while d describes its height. The sample box is used calculate a weighted sum of the grey values in A, that overlap the sample box. The weights are based upon the fraction of area of the intersection between the sample box region and the covered pixel regions.

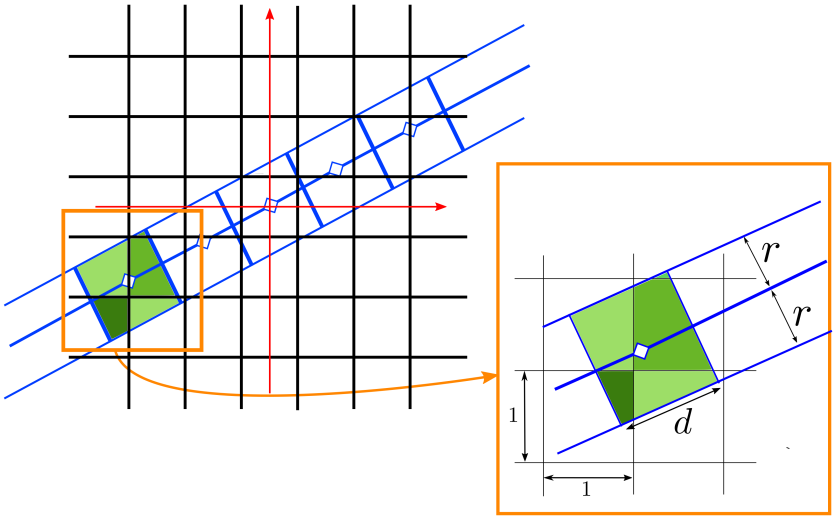


Figure 5.9: Area percentile sampling method.

By making use of the coordinate transformation

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix},$$

the sample box is defined as

$$\mathbb{B}(\vec{c}, r, d, \theta) = \{(u, v) \in \mathcal{A} | u \in [c_u - r, c_u + r], v \in [c_v - \tfrac{d}{2}, c_v + \tfrac{d}{2}]\},$$

where $[c_u, c_v] = [c_x \cos\theta + c_y \sin\theta, c_y \cos\theta - c_x \sin\theta]$. The sample value at $\vec{c} = [c_x, c_y]$ is therefore given by the following equation

$$s = \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{\mathbf{Area}[\mathbb{A}(i,j) \cap \mathbb{B}(c, r, d, \theta)]}{2rd} A_{i,j}. \tag{5.6.1}$$

The embossed function $\mathbb{A}(\cdot)$ describes the pixel region, such that

$$\mathbb{A}(i, j) = \{(x, y) \in \mathcal{A} | x \in [i - \tfrac{1}{2}, i + \tfrac{1}{2}], y \in [j - \tfrac{1}{2}, j + \tfrac{1}{2}]\}.$$

Therefore the term

$$\mathbf{Area}[\mathbb{A}(i, j) \cap \mathbb{B}(c, r, d, \theta)],$$

gives the size of the intersecting area between the pixel region and the sample box. The definitions above describe the theory needed to define the area percentile sampling method, however this does not explain how the areas $\mathbf{Area}[\mathbb{A}(i, j) \cap \mathbb{B}(c, r, d, \theta)]$ can be calculated. We found that the simplest method involves the summation of the area of triangular regions within the

sample box[2].

The triangles are identified by finding all the intersection points between pixel boundaries and the sample box boundaries. The intersecting points, the pixel vertices that are found in the sample box

$$[i + \tfrac{1}{2}, j + \tfrac{1}{2}] \in \mathbb{B}(\vec{c}, r, d),$$

and the four vertices of the sample box describe the desired regions. From here it is a matter of constructing the triangles in the right order, and calculating the sum across all the triangular regions within a pixel.

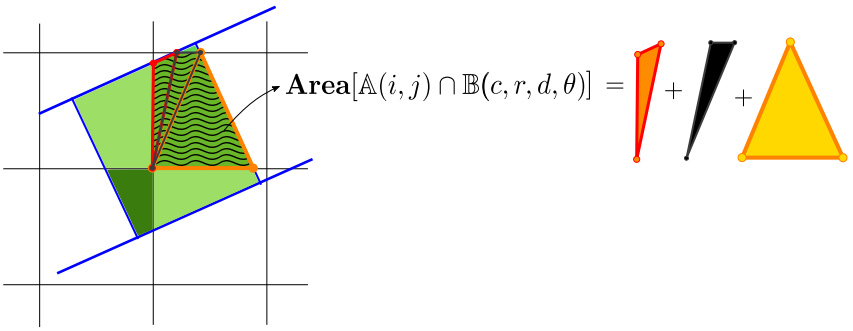


Figure 5.10: Area percentile weight calculation.

5.7 Comparing sampling techniques

The performance of nearest neighbour, bilinear and area percentile sampling methods are compared by sampling three edge profiles of two smooth edges as shown in Figure 5.11. For each sampling method, the profiles are combined to produce a mean sample, and then compared to each other. The edges lie at an angle $\theta = 20°$.

[2]In MATLAB it is even simpler since the POLYAREA function can be used to calculate the area of the polygons directly.

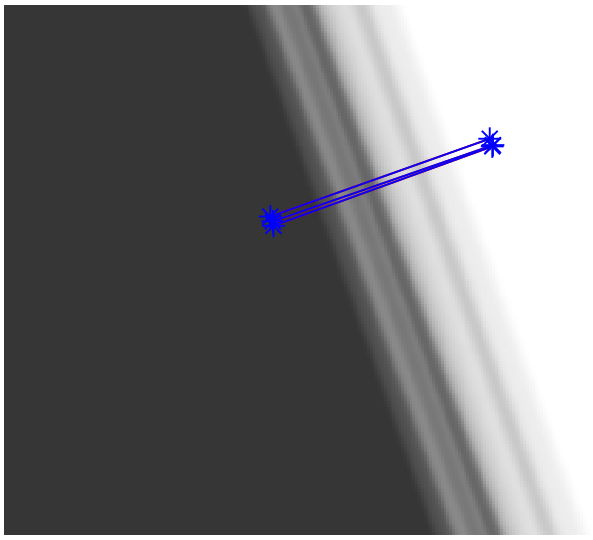


Figure 5.11: Sampling edge profiles in a blurred image B.

The sampling was done at step sizes of $d = 1$ and $d = 0.2$. For Figures 5.12 and 5.13 the sampling techniques are plotted as follows:
nearest neighbour in blue, bilinear in red and area percentile in black. In order to compare the quality of the samples a comparison is also made with the derivatives of the edge profiles.

Edge profile:

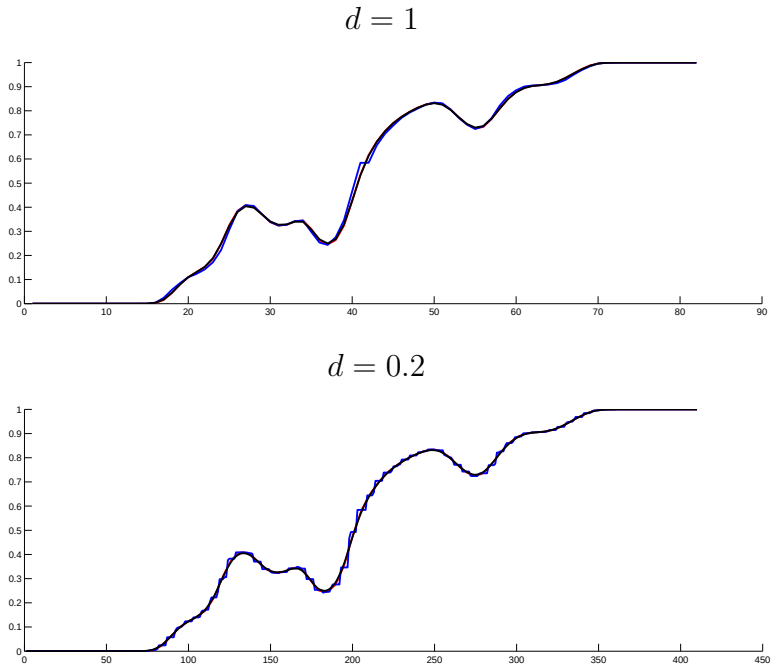


Figure 5.12: Mean of sample edge profiles.

Edge profile derivatives:

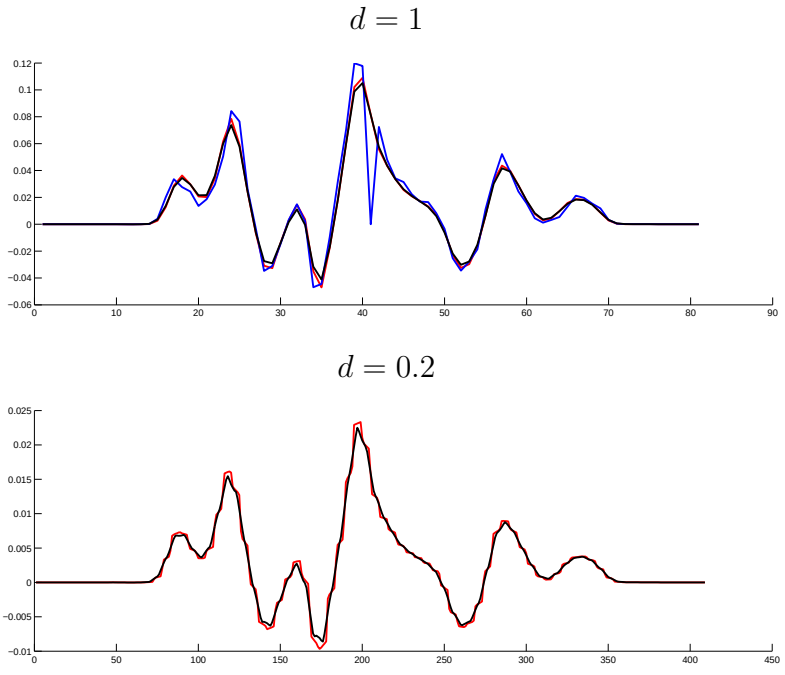


Figure 5.13: Derivatives of mean.

The graphs in Figures 5.12 and 5.13, illustrate that bilinear and area percentile sampling perform better than nearest neighbour sampling. The choice of sampling method, therefore depends on the quality and the smoothness of area percentile compared with that of bilinear sampling.
When investigating the edge and derivative comparisons, it can be seen that the two methods perform almost equivalently for step size of $d = 1$. However when the step size is reduced to $d = 0.2$, its is clearer to see that the area percentile does seem to provide a smoother result.
However, the extra calculations that have to be made when the area percentile is calculated, slows it down considerably in comparison to bilinear sampling. On average it was found that a single bilinear sample was completed in ± 0.035 seconds, whereas the area percentile method took ± 0.0775 per sample. This slight difference in run-time becomes great when considering an image that has hundreds of edges, which are sampled at multiple points along the edge profiles.

5.8 Edge selection

Blurred edge profiles can be found by utilizing edge detection and image sampling. From these a sinogram of the blur kernel will be reconstructed. The edges that are typically found in a blurred image are often perturbed by noise. When adding noise to the blurred image model, a blurred image B is modeled as

$$B(x, y) = A(x, y) * K(x, y) + N(x, y). \tag{5.8.1}$$

Blurred edges often have multiple steps or have been combined with other actual edge that lie close by (if the blur kernel is large enough). The sampling technique that is used might also have an effect on the quality of the edge profile. Therefore identifying "good" edges, those that can successfully be used to obtain a sufficiently smooth sinogram, is important.

5.8.1 Selecting good edges

When looking for good edge profiles it is essential to define the term *"good"*. It is assumed that a blurred edge resulting from the convolution of an ideal step edge and a blur kernel with no noise provides the best edge profile that can be found. That is the special case when a blurred edge image E_B is given by

$$E_B(x, y) = A_\theta(x, y) * K(x, y) \tag{5.8.2}$$

$$\text{where} \quad A_\theta(s, t) = \begin{cases} a_0 & s < 0 \\ a_1 & s > 0 \end{cases}, \quad \text{and} \quad \begin{bmatrix} s \\ t \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}. \tag{5.8.3}$$

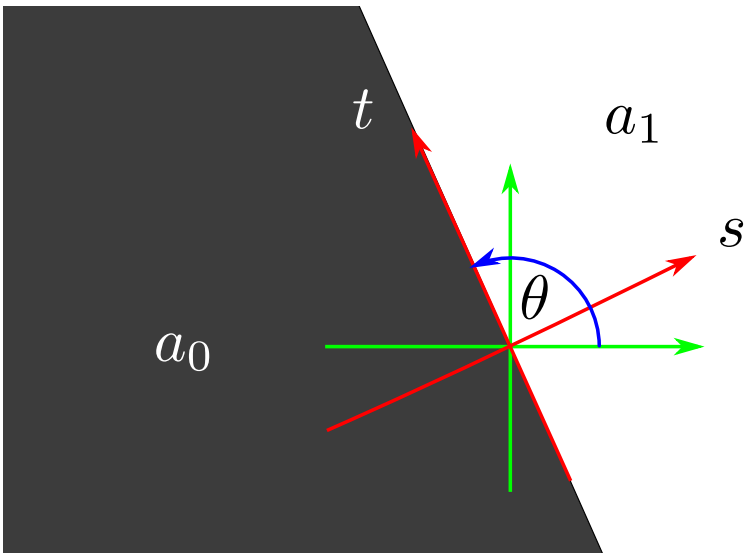


Figure 5.14: Illustration of blurred ideal edges produced by: a kernel with a single mode(red), a kernel with multiple modes(green).

These edges are typically smooth and monotonically increasing, however blur kernels that have more than one mode produce edges with several smaller steps.

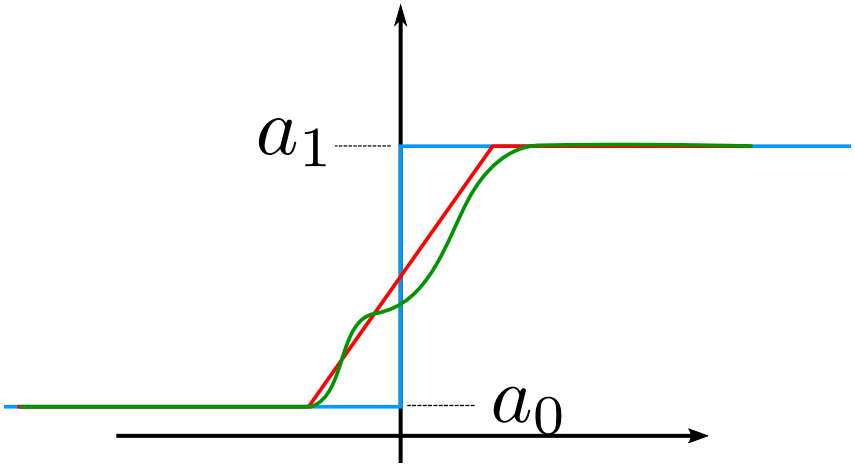


Figure 5.15: Illustration of blurred ideal edges.(blue: The ideal edge; red: ideal edge linearly blurred; green: ideal edge with non-linear blur).

For edges to be accepted as good, they must have similar properties to those found in a blurred ideal step edge example.

5.8.2 Selecting edges

Decision criteria for classifying an edge profile as good, must be set up in accordance with the blurred ideal step edge properties.

It is assumed that a blur kernel is energy conserving, and that all the elements of the kernel are strictly greater than or equal to zero. Further, we have that any sampled edge profile has a size of $2n+1$ elements. These elements are numbered from $-n, \ldots, n$. For an edge profile to be considered good, it must satisfy the statements regarding the following properties of a blurred ideal step edge profile $\vec{e}_B$ at $t = t_c$. The reader may refer to equations (5.8.2), (5.8.3):

- `Monotonically increasing`: The derivative of an increasing step (that is when $a_1 < a_0$) will remain positive after it has been blurred, unless the edge has been affected by noise.

- `Constant intervals of minimum and maximum`: A sampled blurred edge profile $\vec{l}_B$ should contain two intervals of length t

$$\vec{l}_B(j_0) = a_0, \quad \text{for} \quad j_0 = -n, \ldots, -n+t \quad \text{and}$$

$$\vec{l}_B(j_1) = a_1, \quad \text{for} \quad j_1 = n-t, \ldots, n.$$

These statements can be checked by using similar techniques as those employed by Joshi *et al.* (2008).

By following the edge profile, a search for the minimum value a_0 and a maximum value a_1 of the edge profile is done. We shall always sample an edge profile as an increasing function.
A procedure which will:
(1) Check that an edge profile satisfies the blurred ideal step edge properties.
(2) Search for the position j_m and value a_1 of the maximum
is completed as follows:

- `Start at the center of the edge profile` $\vec{l}_B$, `that is when` $j = 0$. `While checking the monotonicity of the edge profile, find the position` $j'_m > 0$ `of the first local maximum.` Use the grey scale value $\vec{l}_B(j'_m)$, as an initial estimate for a_1,

$$a'_1 = \vec{l}_B(j'_m).$$

- `Find the vector` $\vec{l}_{a'_1}$ `by thresholding all elements in` $\vec{l}_B$ `that are not close to` a'_1, that is

$$\vec{l}_{a'_1} = \{\vec{l}_B(j) \mid ||\vec{l}_B(j) - a'_1|| < \epsilon \text{ for } \epsilon = 0.1.\}$$

- Remove all elements of $\vec{l}_{a'_1}$ that are not *directly connected* to the element j'_m. Directly connected refers to those elements $\vec{l}_B(j)$ for which there is no threshold element between $\vec{l}_B(j)$ and $\vec{l}_B(j'_m)$.
- Measure the size of $\vec{l}_{a'_1}$ to check if there is indeed a constant interval for which $\vec{l}_B(j) \approx a_1$.
- Take the mean of $\vec{l}_{a'_1}$ and let this be the final estimate for a_1, that is
$$a_1 = \text{mean}(\vec{l}_{a'_1}).$$
- Update the position of the maximum j_m to the position $j_f > 0$, the first element in $\vec{l}_B$ greater than a_1, that is
$$j_m = j_f \text{ where } \vec{l}_B(j_f) > a_1.$$
- Set all $\vec{l}_B(j) = a_1$ for $j > j_m$.

A similar technique is used when obtaining the minimum a_0.

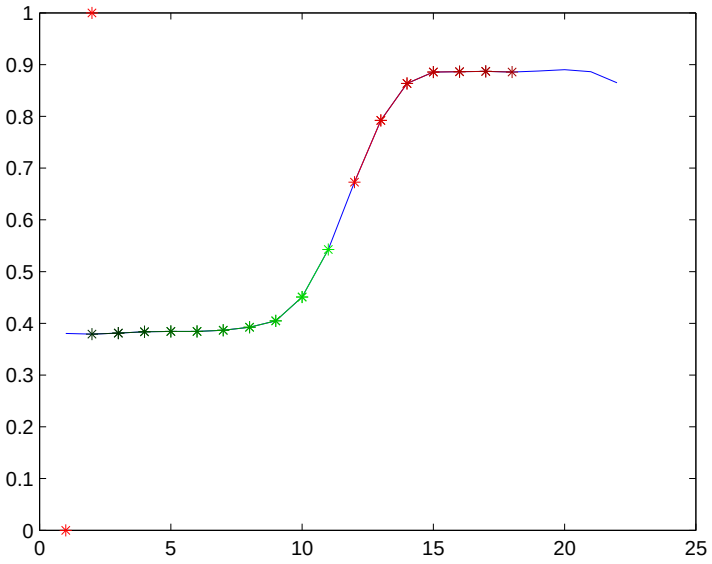


Figure 5.16: Following the edge profile.

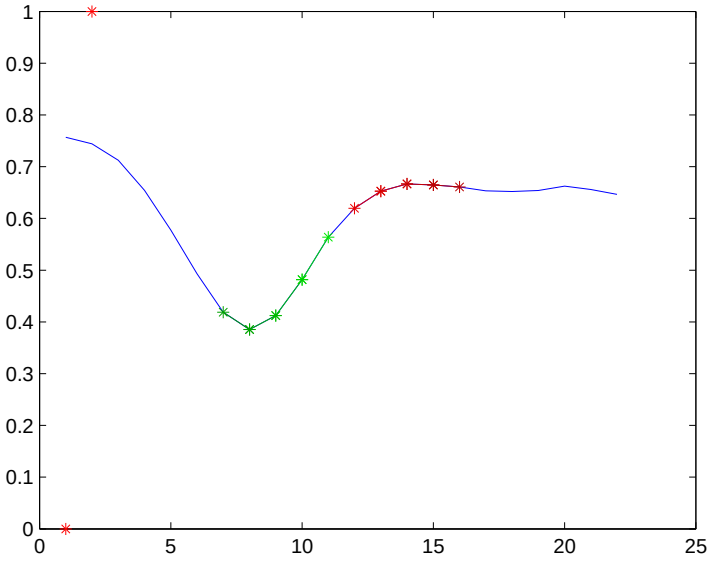


Figure 5.17: A profile that does not satisfy the conditions of a good edge.

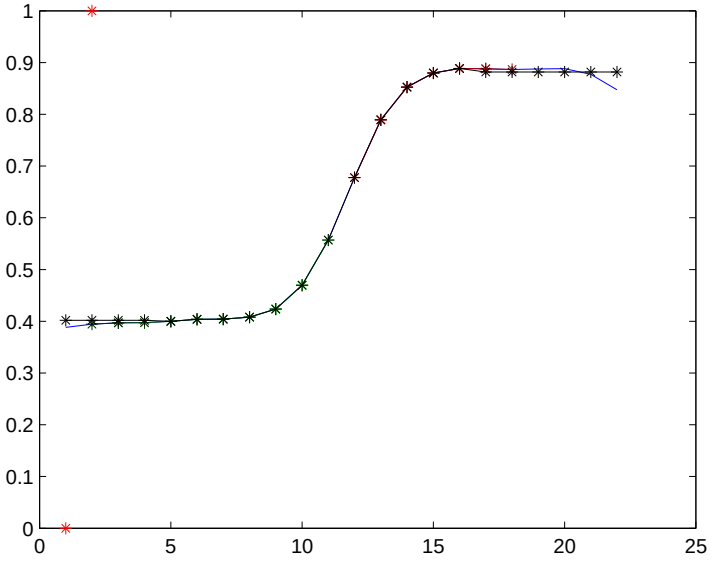


Figure 5.18: A successfully sampled edge.

5.9 Improving edge quality

Good edges that are found in an image may still contain noise and are therefore not necessarily suitable to be used in the sinogram. However there are many ways to improve the quality of the edges that are used in the sinogram. Consider the following example in Figure 5.19:

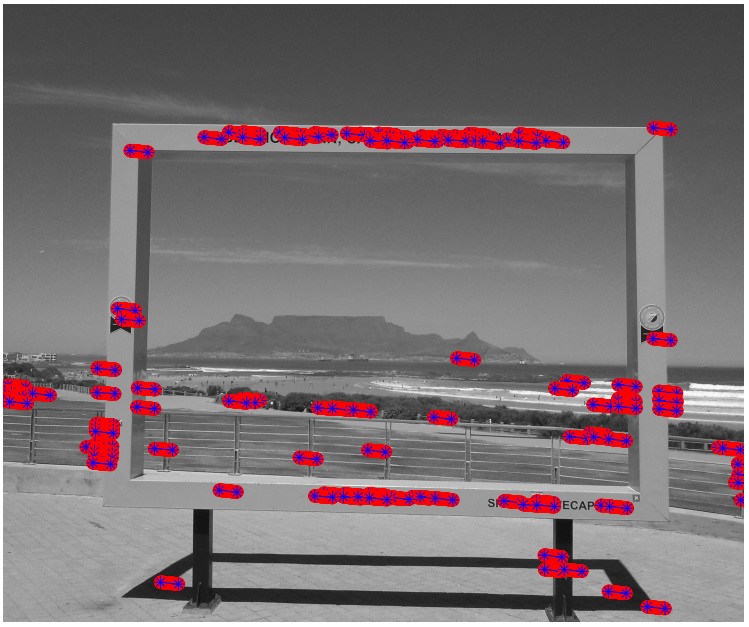


Figure 5.19: Multiple edges found at the same angle.

This shows an that image contains multiple edges that lie at the same angle. Statistical analysis of the edge profiles is used to produce an improved sinogram column, if a large enough number of edges are successfully sampled in an blurred image.

Definition 5.1. Let $\mathcal{L}(\theta)$ be the set of all good edges profiles in a blurred image B, that have been sampled at an angle θ, such that

$$\mathcal{L}(\theta) = \{\vec{l}_{-k}, \ldots, \vec{l}_k\}.$$

Each $\vec{l}_h$ is a column vector for $h = -k, \ldots, k,$.

The profiles that are found in $\mathcal{L}(\theta)$ will be used to construct a single sinogram column S_θ. The quality of a sinogram column S_θ can be improved by taking the mean of the set $S_\theta = \overline{\mathcal{L}(\theta)}$. However the mean $\overline{\mathcal{L}(\theta)}$ can also be improved by removing outliers.

5.9.1 Centering edges

Before the statistics of the edges can be used to produce improved sinogram columns must be contented correctly. This is done to avoid destroying important edge information that is caused by edges being found off center. Take the following example:

Figure 5.20: An image with sampled edges.

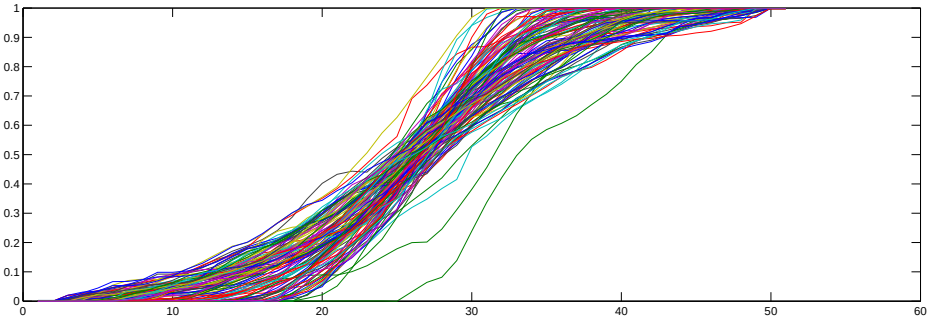


Figure 5.21: Good edges centered incorrectly.

To rectify this, the edge profiles, see Figure 5.21 are centered according to their *first moment*. The position of the first moment of an edges is found as follows,

$$f_m = \sum_{h=-k}^{k} \frac{h\vec{l_B}(h)}{2k+1}.$$

Once the position of the first moment has been calculated, the edge can be correctly centered by shifting mid point to the first moment position, see Figure5.22.

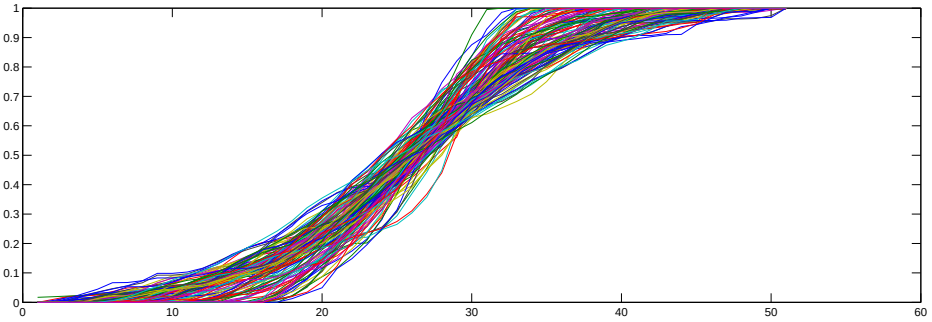


Figure 5.22: The sampled edges centered correctly.

5.9.2 Standard deviation method

The edge quality can be improved by removing outliers . This is done by calculating the mean of the edge profile set $\mathcal{L}(\theta)$. An edge $\vec{l_h}$ is removed if its error value is smaller than a confidence value ϵ, according to the error estimate

$$||\overline{\mathcal{L}(\theta)} - \vec{l_h}|| < \epsilon.$$

A good measure for ϵ is given by the norm of the standard deviation of edge profile set. Figure 5.23 shows the inlier set found for the test image, see Figure 5.20.

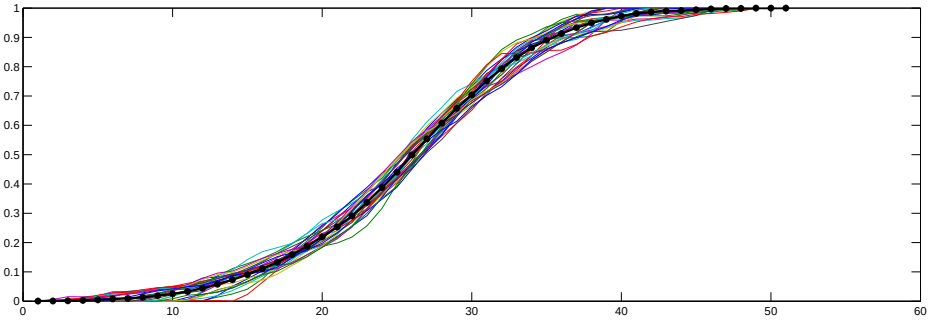


Figure 5.23: Outliers removed.

5.9.3 RANSAC

RANSAC is an algorithm that was proposed by Fischler and Bolles (1981). It is a general parameter estimation approach, and is used in many computer vision applications. Here it has been adapted to improve edge selection and remove bad edges. The RANSAC algorithm is initialized with an empty set of inliers. RANSAC then repeats the following steps for some i number of iterations:

1. `Select a random subset from the edge data set` $\mathcal{L}(\theta)$. Call this subset the hypothetical inliers.

2. `Fit a model to the set of hypothetical inliers.` This is calculated as the mean of all hypothetical inliers, call it $\vec{l}_{\text{in}}$

3. `Test all other edges in` $\mathcal{L}(\theta)$ `and find the total number of inliers.` This is done by finding all edges that lie within an confidence level ϵ such that
$$||\vec{l}_{\text{in}} - \vec{l}_m|| < \epsilon.$$

4. `Check size of inlier set.` If the newly estimated set of inlier is larger than previously estimated, make it the current set of inliers.

All edges that fall outside of the inlier set are then removed from $\mathcal{L}(\theta)$. The mean is recalculated to obtain an improved sinogram column. Figure 5.24 shows the inlier set found using RANSAC for the test image, see Figure 5.20.

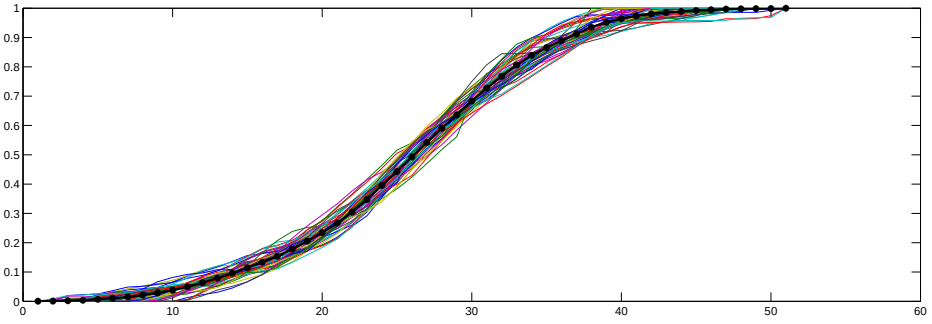


Figure 5.24: Outliers removed using RANSAC.

Chapter 6

Tomographic estimation and sinogram interpolation

6.1 The direct inversion algorithm

The theory presented thus far enables us to implement *the direct inversion algorithm*, as referred to by Cho *et al.* (2011). The direct inversion algorithm, as its name suggests, will obtain the blur kernel projection data from the blurred image and process the data as shown in chapter 5. The projection data is then directly inverted, by using the inverse Radon transform, to obtain the image blur kernel. We briefly present some initial results obtained using the direct inversion algorithm.

Example A

Figure 6.1: A blurred test image and deblurred test image using the kernel estimated using direct inversion from a sparse sinogram.

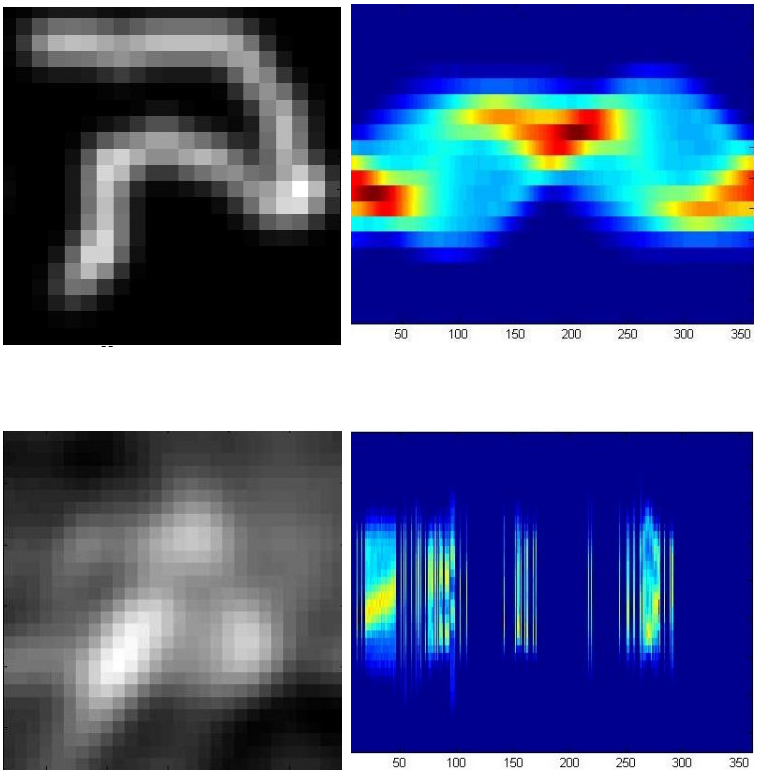


Figure 6.2: Failed blur kernel estimation. Shown is the ground truth blur kernel and its sinogram, the estimated blur kernel (using direct inversion) and its sparse estimated kernel sinogram.

The results from Figure 6.2 show that the Radon transform can be used to find radial information about the blur kernel. However they also highlight a severe shortcoming of the direct inversion algorithm.
The blur kernel can only be estimated with a high enough quality if a sufficient number of edges are found in the image A at various angles. Figure 6.1 shows a test image that was blurred by the blur kernel, shown in Figure 6.2. The blurred test image was sampled and a random sparse sinogram of the blur kernel was obtained. The blur kernel was reconstructed from the sparse sinogram and used to deblur the test image. From Figure 6.1 it can be seen that the blur kernel was not accurately estimated, hence the resulting deblurred image is of low quality.

6.2 The artifact problem

The problem lies with the inversion of the sparse and random sinogram which is obtained from the edge sampling process in chapter 5[1].

[1]The `MATLAB` function `IRADON` uses filtered back projection to invert an image sinogram.

The reconstructed test image and sparse sinogram.

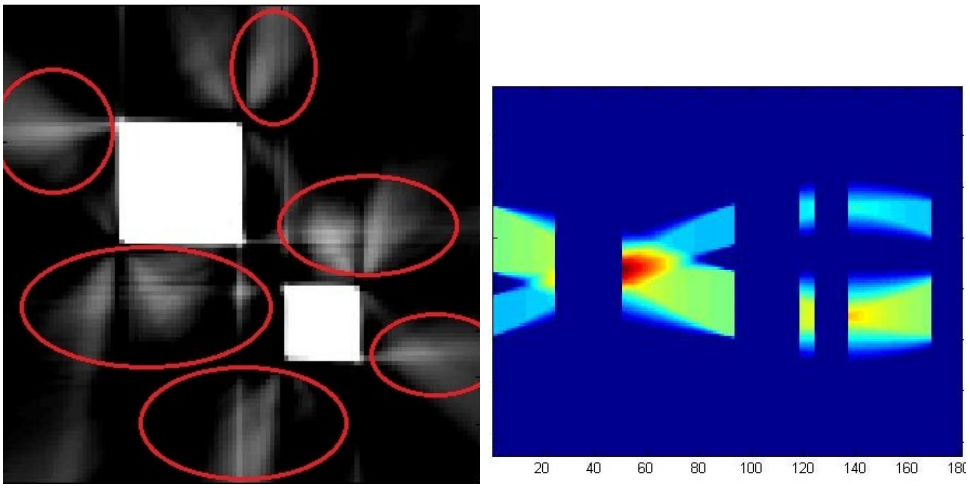


Figure 6.3: Illustration of projection artifacts: The test image reconstructed from a sparse sinogram, it contains projection artifacts circled in red.

These artifacts occur because the sinogram contains no projection data in a particular direction to cancel out the back projection data from an obtained sinogram column. When estimating blur kernels these artifacts are unwanted since they alter the blur kernel. This problem is also identified by Cho *et al.* (2011) in order to correct this they make use of a MAP search, that fits an estimated blur kernel to the obtained kernel projection data.
In order to retain the non-parametric and direct nature of the tomographic algorithm we propose that a *sinogram interpolation* technique as described in Kalke and Siltanen (2014) be used. A sinogram interpolation technique will estimate the missing sinogram columns and when used in filtered back projection, and will cancel out the projection data that cause the projection artifacts to occur. Sinogram interpolation estimates the missing sinogram columns directly from the obtained blurred image sinogram, while retaining the original sinogram data.

6.3 Sinogram interpolation

Notation

For the section on sinogram interpolation the following notation scheme will be used.

Continuous indices:

Notation	Description	Range
x, y	Position in a continuous image.	$x, y \in (-\infty, \infty)$
θ	Projection angle for the forward radon transform.	$\theta \in [0, 2\pi]$
r, ϕ	Position in a continuous image in polar coordinates.	$\phi \in [0, 2\pi], \quad r \in [0, \infty)$

Discrete indices:

Notation	Description	Range
i, j	Pixel positions in a discrete image.	$i = 1, \ldots, n$, $j = 1, \ldots, m$
t	Index of the t-th sinogram column at angle $\theta_t = t\Delta\theta$.	$\theta_t \in [0, 2\pi]$, $t = 1, \ldots, \bar{\theta}$
r_k, ϕ_k	Pixel positions in a discrete image in polar coordinates.	$\phi \in [0, 2\pi]$, $r \in [0, \sqrt{(n/2)^2 + (n/2)^2}]$
i_c, j_c	Image center.	$i = \frac{1}{2}n, \quad j = \frac{1}{2}m$
h	Index for sinogram row.	$h = -\frac{1}{2}\bar{h}, \ldots, \frac{1}{2}\bar{h}$.

Clarification of values:

Value	Description
A	The original image in Cartesian coordinates.
V	The original image in polar coordinates.
S	The sinogram of A or V.
W	The set of all warps.
n, m	Original image size.
$\bar{h}$	Number of sinogram rows.
$\bar{\theta}$	Number of sinogram columns.
$\bar{k}$	Number of warps found in a known sinogram.
$\bar{L}$	Number of warps that pass through a known sinogram element.

6.3.1 Sinograms and the forward Radon transform

A technique for improving the obtained blur kernel sinogram is presented here. Methods developed by Kalke and Siltanen (2014) for sinogram interpolation are used as a basis for the work that follows.

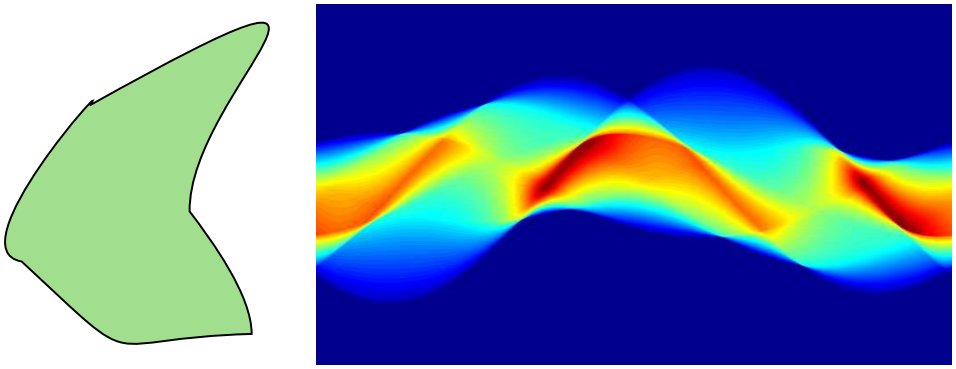


Figure 6.4: An image and its sinogram

By definition of the forward Radon transform Toft (1996), a continuous representation of sinogram column $\mathbf{s}_{\theta_t}$ can be expressed as the line integral over the line[2] $\hat{h} = x\cos(\theta) + y\sin(\theta)$

$$\mathbf{s}_\theta(\hat{h}) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} A(x,y)\delta(\hat{h} - x\cos(\theta) - y\sin(\theta))dx\,dy. \tag{6.3.1}$$

[2]We choose to use this definition of the line integral since here δ has the geometric interpretation of allowing only points that fall on the line $\hat{h}$ to be include in the summation for $\mathbf{s}_{h,\theta}(\hat{h})$. In the discrete equation δ corresponds to the chosen method of image sampling.

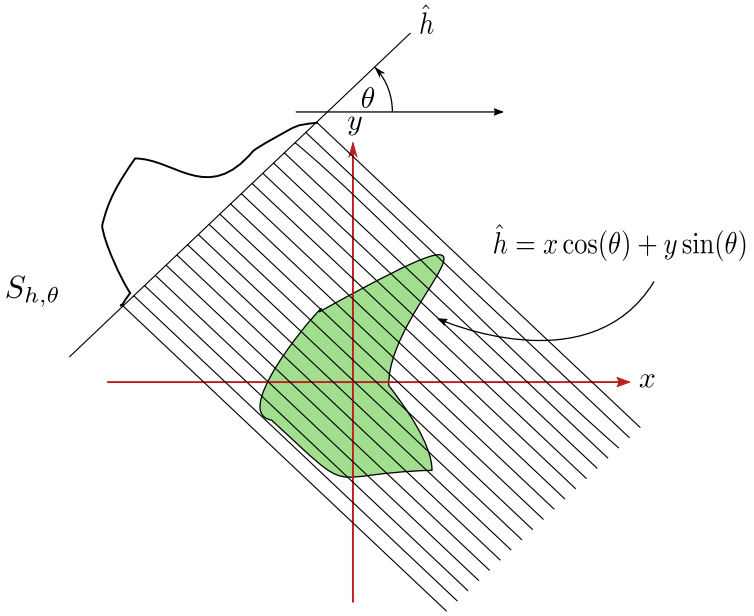


Figure 6.5: The forward Radon transform

We aim to describe (6.3.1) using the polar coordinate transformation:

$$x = r\cos(\phi), \quad y = r\sin(\phi) \quad \text{where} \quad dx\,dy = r\,dr\,d\phi.$$

Applying this transformation to the original image $A(x, y)$ according to equation (6.3.1), we relabel $A(x, y)$ as

$$V(r, \phi) = A(x, y).$$

Then

$$\mathbf{s}_\theta(\hat{h}) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} V(r,\phi)\delta(\hat{h} - r\cos(\phi)\cos(\theta) - r\sin(\phi)\sin(\theta))r\,dr\,d\phi \quad (6.3.2)$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} V(r,\phi)\delta(\hat{h} - r\cos(\phi+\theta))r\,dr\,d\phi. \quad (6.3.3)$$

The last expression represents the integral over the line

$$\hat{h} = r\cos(\phi + \theta). \quad (6.3.4)$$

A discrete sinogram is a matrix S with column vectors $\mathbf{s}_\theta$,

$$S = [\mathbf{s}_{\theta_1}, \mathbf{s}_{\theta_2}, \ldots, \mathbf{s}_{\theta_{\bar{\theta}}}],$$

where each column $\mathbf{s}_{\theta_t}$ is the Radon projection of an image $A(x, y)$ at the angle θ_t. The angles are stored in the vector

$$\Theta = [\theta_1, \theta_2, \ldots, \theta_{\bar{\theta}}].$$

The polar form of the Radon transform (6.3.3) is discretized by letting

$$x_j = j\Delta x, \quad \text{for } j = 1, \ldots, m,$$
$$y_i = i\Delta y, \quad \text{for } i = 1, \ldots, n,$$
$$A_{i,j} = A(x_j, y_i), \quad \text{with}$$
$$\theta_t = t\Delta\theta, \quad \text{for } t = 1, \ldots, \bar{\theta}.$$

In practice, when discretizing (6.3.1) a sinogram element is expressed as

$$\mathbf{s}_{\theta_t}(\hat{h}) = \sum_i \sum_j A_{i,j}\Delta(\hat{h} - x_j\cos(\theta_t) - y_i\sin(\theta_t)), \tag{6.3.5}$$

which simply means that each element of the sinogram S_{h,θ_t} is the discrete summation over all the pixel values along the line $\hat{h} = x\cos(\theta_t) + y\sin(\theta_t)$. Here $\Delta(\cdot)$ is a "selection function" corresponding to the users choice of discrete sampling method over continuous line $\hat{h}$.

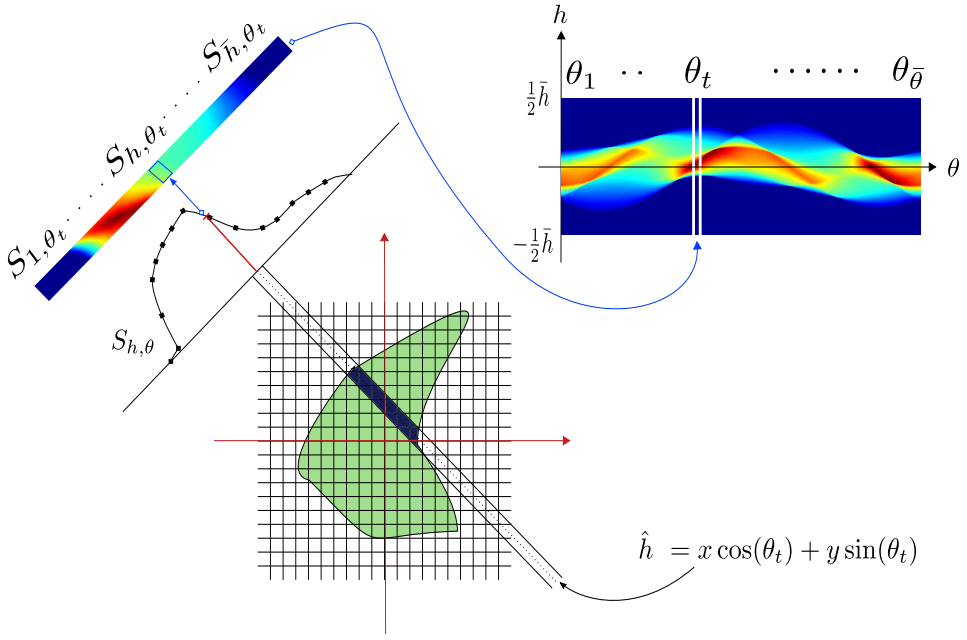


Figure 6.6: Interpretation of summation for discrete forward transform

Note that when dealing with the polar coordinates, a different indexing method is used.Using equation (6.3.3) along with

$$r_k = k\Delta r, \text{ and } \phi_k = k\Delta\phi \quad \text{for } k = 1, \ldots, (mn),$$

a sinogram element is expressed as

$$\mathbf{s}_{\theta_t}(\hat{h}) = \sum_k V(r_k, \phi_k)\Delta(\hat{h} - r_k\cos(\phi_k + \theta_t))r_k. \tag{6.3.6}$$

This is equivalent to equation (6.3.5), since the summation is still performed over points along $\hat{h}$. However, the points along $\hat{h}$ are now expressed in the polar coordinates (r, ϕ) by

$$u_k = r_k \cos(\phi_k + \theta_t),$$

called the *pixel position curve*. This equation corresponds to a cosine curve that passes through the sinogram for each pixel in the image A. This implies that the sinogram of an image can be calculated by finding the position curves for all the pixels in the image and then performing a summation over the grey scale values, see Figure 6.3.1.

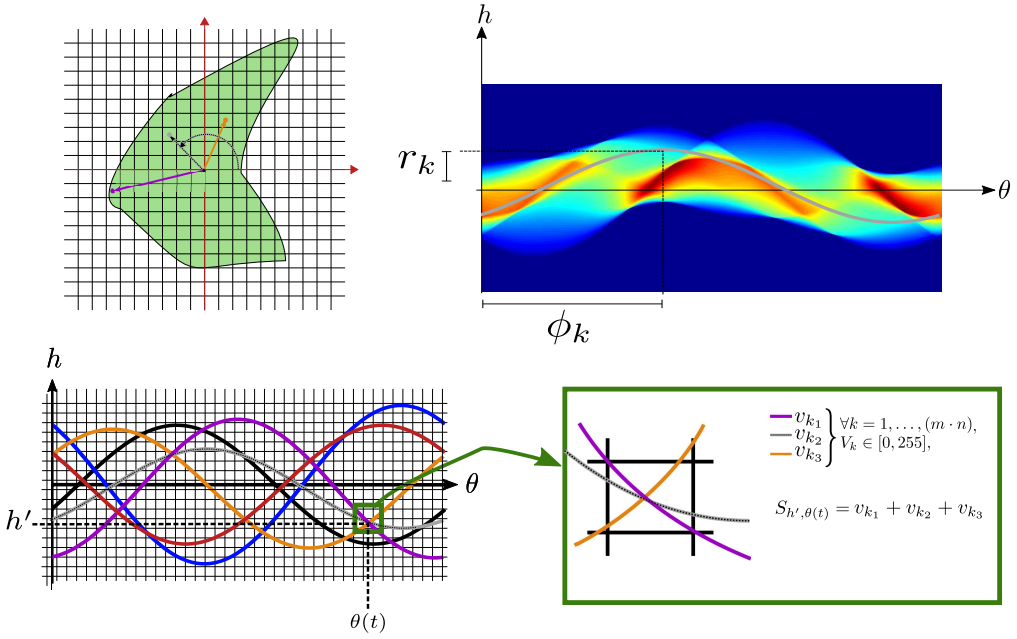


Figure 6.7: Relating pixel position curves to the image sinogram.

By combining the image grey scale value and pixel position curve we will define the concept of a sinogram *warp*. The warp will be used as the basis for the sinogram interpolation procedure. The term warp was first used by **?**.

6.3.2 Warps

A *warp* w_k is a weight carrying cosine wave that passes through only strictly positive elements in a sinogram.

$$w_k = \{v_k, u_k(\theta)\}, \quad w_k \in W$$

where v_k is the warp value, and

$$u_k(\theta) = r_k \cos(\theta + \phi_k) \tag{6.3.7}$$

is referred to as a warp curve with r_k the warp amplitude and ϕ_k the warp phase angle.

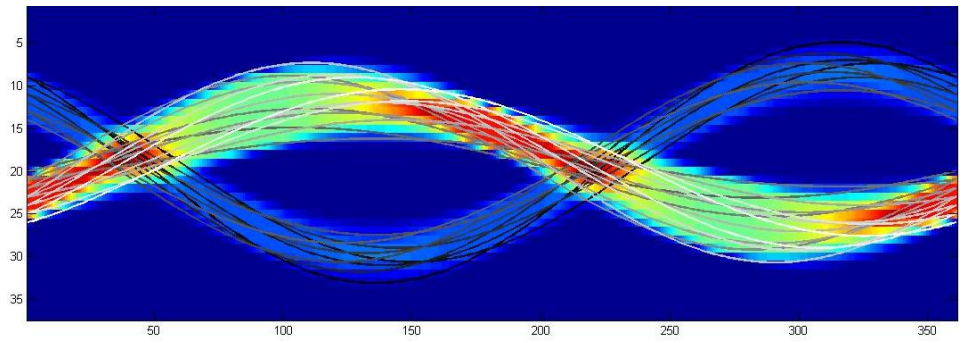


Figure 6.8: Warps passing through a sinogram

For such a curve to be defined as a warp, the original image and its given sparse sinogram, as well as each curve that passes through that sinogram must satisfy the following *warp conditions*:

1. Each non-zero pixel in the original image produces exactly one warp.

2. Each warp intersects only non-zero elements in the sinogram, equivalently all warps carry a weight that is greater than zero

$$v_k > 0.$$

3. Each non-zero element in the sinogram is the linear combination of all the warps that pass through that point, that is:

$$S_{h,\theta} = \sum_{k=1}^{L} c_k v_k$$

where L is the number of warps that pass through the sinogram element $S_{h,\theta}$.

4. The sum of all warps is equal to the sum of a single sinogram column, and is also equal to the sum of all elements in the original image,

$$\sum_{h=1}^{\bar{h}} S_{h,\theta_t} = \sum_{k=1}^{\bar{k}} v_k = \sum_{i} \sum_{j} A_{i,j},$$

where $\bar{k}$ is the total number of warps.

The general expression for the warp curve (6.3.7) can easily be derived by the graphical representation of condition (1).

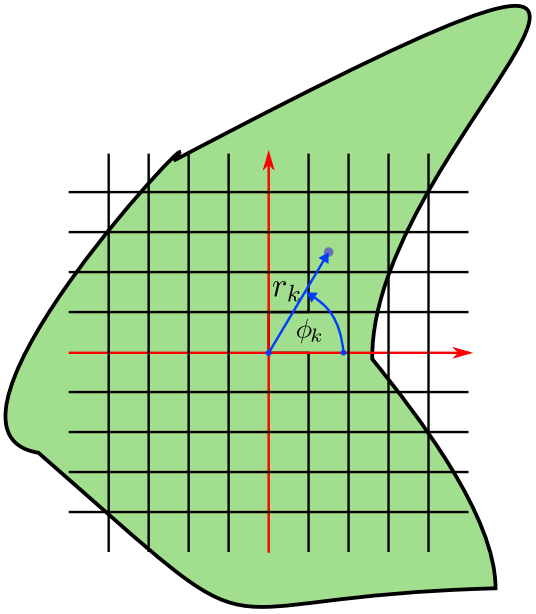


Figure 6.9: Defining the warp curve from the original image

From Figure 6.3.2 it can be shown that

$$r_k = \sqrt{(i_c - i)^2 + (j_c - j)^2}, \tag{6.3.8}$$

$$\phi_k = \arctan\left(\frac{i_c - i}{j_c - j}\right), \tag{6.3.9}$$

where (i_c, j_c) is the center point of the original image and (i, j) the current pixel location.

In order to interpolate a sinogram the warps are used in an inverse sense[3], by utilizing the known sinogram columns from a sparse sinogram the warp curves can be found from a pseudo image[4]. The pseudo image only specifies if a pixel from the original image was indeed an positive element, which would imply that the warp has a positive weight. Once all warp curves have been found for a sinogram, it is left to determine their weights. The warp weights are obtained by setting up and solving the equations provided by conditions (2) to (4) as a non-negative least squares approximation. For our purposes we let $c_k = 1$ for all k.

[3]This is similar to the inverse procedure used when rotating or scaling an image.

[4]This image is not actually constructed but the warp curves are identified by iterating through the rows and columns of the pseudo image.

The elements of a required sinogram column $S_{\theta'}$ are then obtained by identifying the positions of all the warps for θ' via the warp curves. Condition 3 is then used to sum the warp weights for each column element, which yields the required sinogram column.

6.3.3 The algorithm

The steps needed to implement the sinogram interpolation algorithm are as follows:

1. `Calculate the original(or maximum image radius) image size.` This corresponds to $\bar{h}\sqrt{2}$ and is taken to be $n = \text{ceil}(\bar{h}\sqrt{2})$. This is done to ensure we match the sampling rate of the Radon transform, which is equivalent to our method of edge sampling.

2. `By making use of equations (6.3.8) and (6.3.9), loop through the original image positions and calculate all the warp phase angles and amplitudes`[5].

3. `For each warp find all the known sinogram points that the warp curve passes through.` This is done by calculating u_k for each known θ of[6] S. The sinogram values at these points are expressed as

$$\mathcal{S} = S_{u_k(\theta),\theta} \text{ for all } \theta \in \Theta$$

with the points $(u_k(\theta), \theta)$ being called the warp points.

   a) `Check that all values of` $\mathcal{S}$ `satisfy condition (2).` If so, catalog all the warp points and assign the warp to a valid warp set.

4. `Setup the system of equations` $X\underline{v} = \underline{\sigma}$ `corresponding to conditions (3) and ((4).` This can be done by looping through the known sinogram S as follows:

   a) `For each sinogram element` $S_{h,\theta(t)}$ `check that` $S_{h,\theta(t)} > 0$ `and that at least one valid warp passes through the point` $(h, \theta(t))$.

   b) `Append` $S_{h,\theta(t)}$ `to` $\underline{\sigma}$`, if the previous step holds true.`

   c) `Simultaneously append a row to the matrix` X `with a one at column` j `if the warp` w_j `passes through the point` $(h, \theta(t))$.

[5] We do not actually recreate the original image but only loop through the relevant pixel positions.

[6] Note these points are off set by $(N+1)/2$ when implementing the algorithm, this is an implementation requirement.

Therefore each column of X represents a single valid warp and should be of size $\bar{s} \times \bar{k}$ where $\bar{s}$ is the number of positive sinogram elements that a warp passes through[7] and $\bar{k}$ the total number of warps.

5. Solve the system $X\underline{v} = \underline{\sigma}$ by finding the non-negative least squares approximation for $\underline{v}$ and therefore obtain an estimate of the warp values v_k.

6. Construct a new sinogram column $S_{\theta'}$ by calculating all valid warp points for the angle θ'. An element $S_{h,\theta'}$ is found by summing the warp values v_k for all warps that pass within a half pixel radius of (h, θ').

6.4 Sinogram interpolation results

The sinogram interpolation results were produced as follows :
A test kernel was created using the motion blur model and its sinogram obtained for $\theta = [0^o, 1^o, 2^o, \ldots, 180^o]$. A random number of sinogram columns (either $10, 20$ or 40) were kept to produce a random sparse sinogram of the test kernel. The random sparse sinogram was then interpolated and both the interpolated as well as sparse sinograms inverted using the `IRADON` in `MATLAB`. The results show the original kernel and sinogram, followed by the inverted kernel and sparse sinogram, followed by the inverted kernel and its interpolated sinogram. The tests were run for two different motion blur kernels (A and B)[8].

[7] This may possibly be less that the total number of positive non-zero elements in the sinogram.

[8] Results were also produced for regularly spaced sinogram columns. A full version of all results are found in appendix B

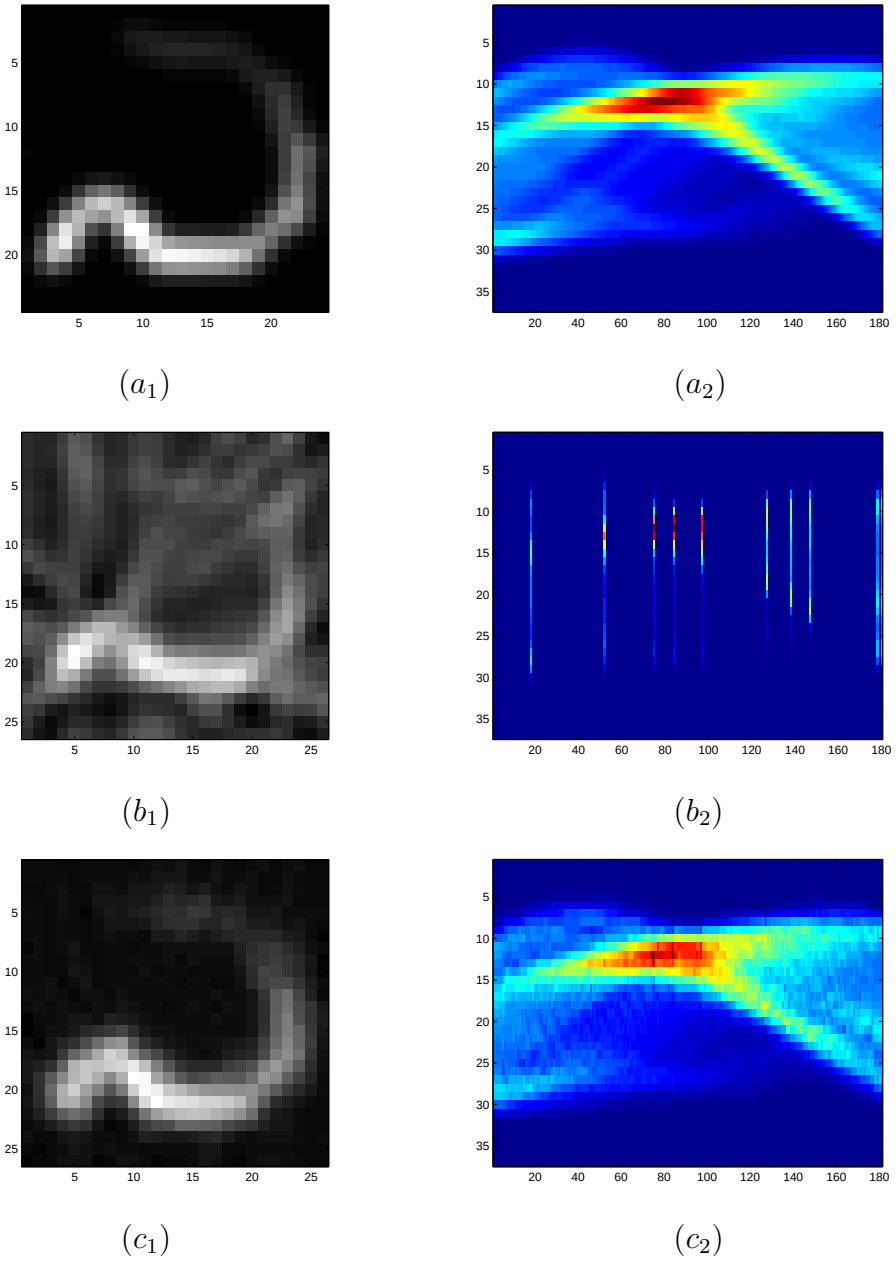


Figure 6.10: Ten sparse random sinogram columns from test kernel A. The simulated blur kernel and its sinogram (a_1, a_2), the sparse sinogram and the recovered kernel(b_1, b_2), the interpolated sinogram and recovered kernel(c_1, c_2).

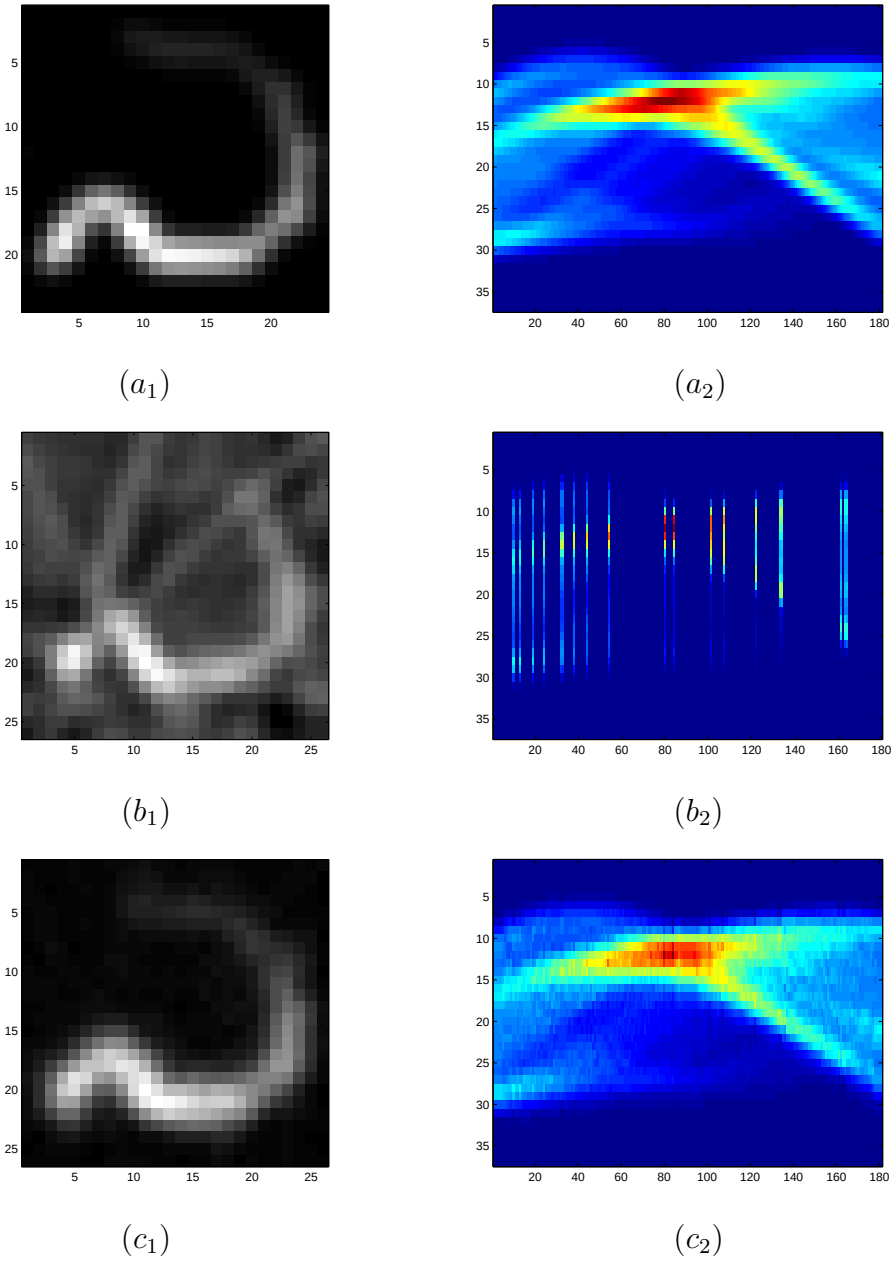


Figure 6.11: Twenty sparse random sinogram columns from test kernel B. The simulated blur kernel and its sinogram (a_1, a_2), the sparse sinogram and the recovered kernel(b_1, b_2), the interpolated sinogram and recovered kernel(c_1, c_2).

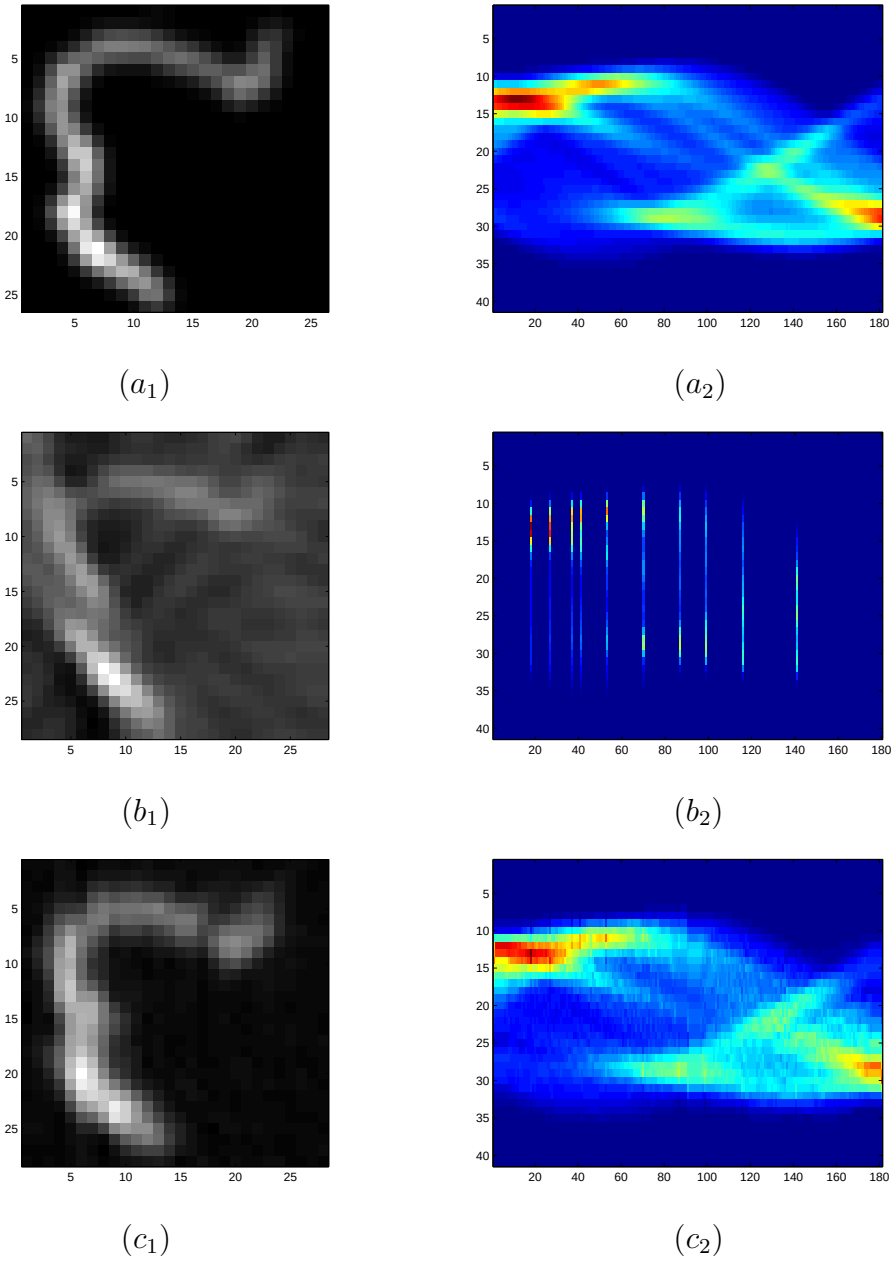


Figure 6.12: Ten sparse random sinogram columns from test kernel B. The simulated blur kernel and its sinogram (a_1, a_2), the sparse sinogram and the recovered kernel(b_1, b_2), the interpolated sinogram and recovered kernel(c_1, c_2).

6.5 Tomographic estimation and sinogram interpolation

The necessary theoretical background for both the tomographic kernel estimation technique and sinogram interpolation has been covered up to this point. Sinogram interpolation will now be used to improve the tomographic kernel estimation technique. We briefly show an overview of how the tomographic kernel estimation technique is updated. With the addition of sinogram interpolation the tomographic algorithm is improved by adding a step which interpolates the sinogram obtained in step 4.

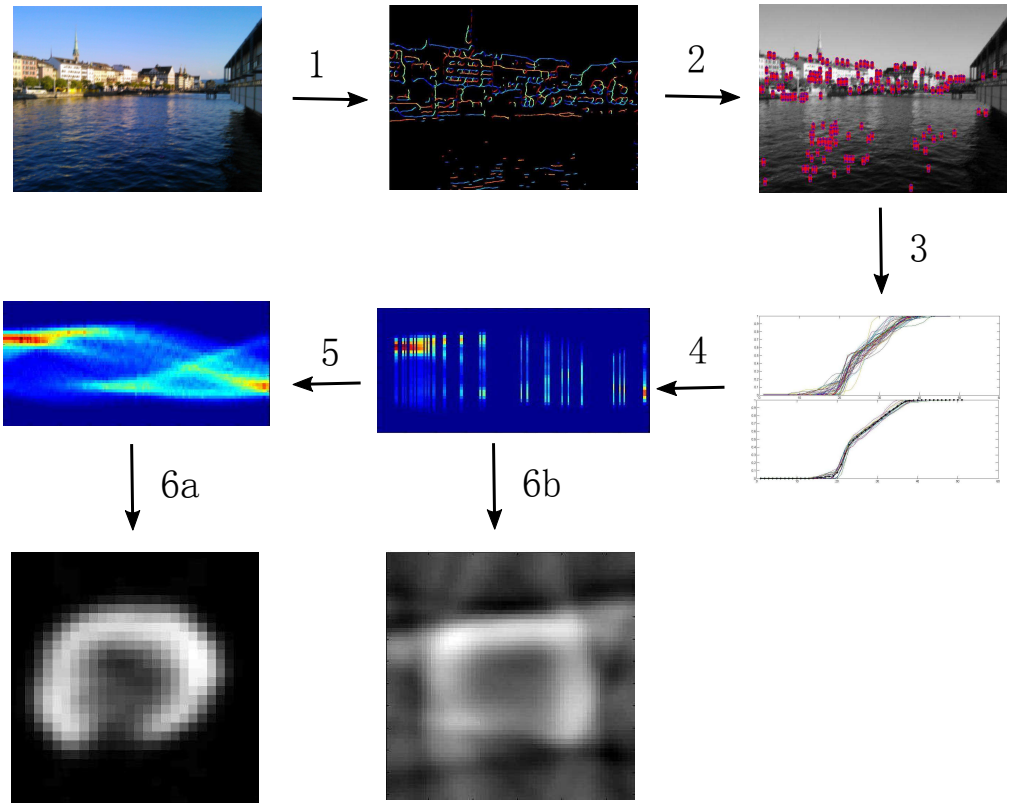


Figure 6.13: The tomographic estimation algorithm.

1. Find the orientation and location of edges in blurred image $B(x, y)$.
2. Sample all edges for $\theta = i,\ i \in [-\pi, \pi]$.
3. Take mean of all samples to produce mean edge profile call it e_i.
4. Take the derivative of e_i for each i to produce a sinogram $S(h, \theta)$ of the blur kernel $K(x, y)$.

5. `Improve the sinogram` $S(h, \theta)$ `by utilizing sinogram interpolation.`

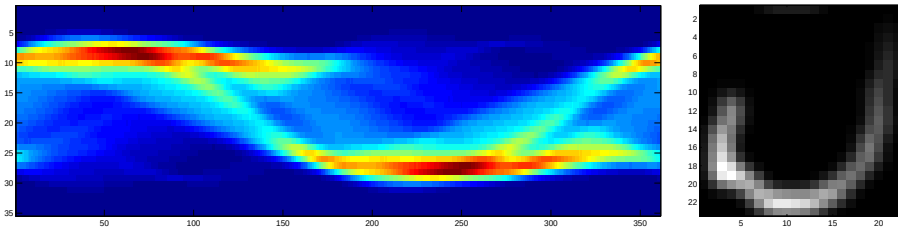
6. `Reconstruct the blur kernel from the improved sinogram using an inverse Radon transform.`

6.5.1 Initial results

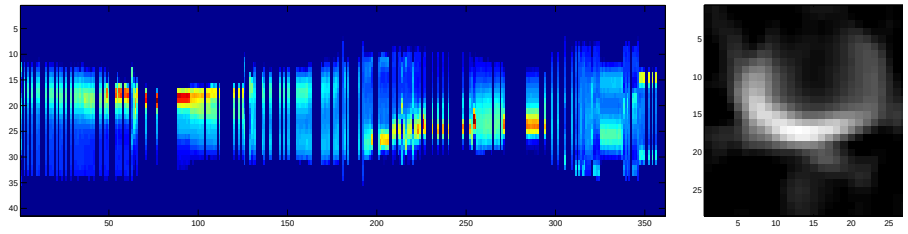
Typical results are shown for the tomographic kernel estimation technique in combination with sinogram interpolation. The purpose of this initial results is to show case the complete blur kernel estimation procedure, followed by image deblurring with the obtained blur kernel.

Figure 6.14: The original test image and blurred test image

(A)



(B)



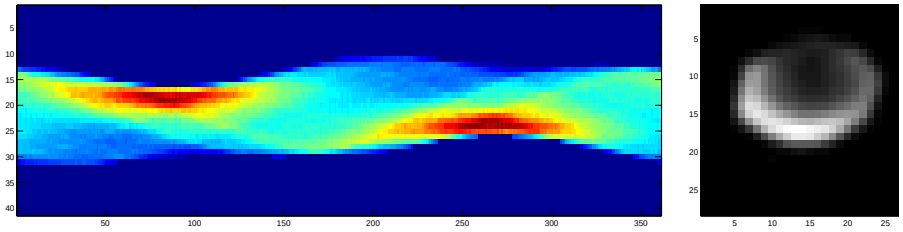
(C) 

Figure 6.15: Sinograms and blur kernels, (A) the original blur kernel and its sinogram, (B) the sampled sinogram and resulting blur kernel and (C) the interpolated sinogram and the resulting blur kernel.

(A) 
(B) 

(C) 
(D) 

Figure 6.16: Zoomed image regions, (A) the blurred image, (B) deblurred using the original blur kernel, (C) deblurred with the direct inversion blur kernel, (D) deblurred with the blur kernel obtained form interpolated sinogram.

Chapter 7

Results and discussion

7.1 Results and discussion

In this chapter we will show and discuss some results obtained from our implementation of the tomographic estimation algorithm in combination with sinogram interpolation.
For all the colour test photographs, edges were sampled[1] from the grey scale converted image at a step size of either 0.5 or 1 depending on the estimated size of the blur kernel. The profiles were grouped into 360 distinct bins from which sinogram columns were estimated. This implies that, the bin sizes range from $[\theta - n, \theta + n]$ where n was chosen to be one degree. The bin $[\theta - n, \theta + n]$ will be referred to as the sample bin at θ with n being the sample range. The sampled sinograms were then interpolated to produce sinograms that span $0^o - 360^o$ and the blur kernel was reconstructed. Thereafter, the blurred photograph colour channels were each deconvolved using the Lucy-Richardson algorithm for ten iterations.

The results are split into two sections, the first will contain results obtained from actual blurred photographs. This is done to confirm that the tomographic algorithm with sinogram interpolation can be used to estimate actual blur kernels and can be used to deblur real world photographs. The second section will compare kernels and deblurred photographs from the tomographic algorithm to those obtained using an implementation of the algorithm described by Fegus *et al.* (2006). Varying photographs and their resulting blur kernel estimates are shown where they are relevant, further results and notes on how various

[1] Testing showed that even though the area percentile sampling method can produce better sample values it was too computationally expensive and had a long run time. It was therefore decided that all the results produced would be done by using the bilinear sampling method which produced satisfactory edge profiles with little computational cost.

kernel estimates were found will be included in Appendix C. Blur kernel point source artifacts can often be found in the blurred image, these artifacts give an indication as to what the true blur kernel looks like. These artifacts are also used as a reference of the true blur kernel by Fegus *et al.* (2006) and will be shown where relevant.

7.1.1 Naturally blurred photographs

Result: Test photograph 1 In this case we will illustrate that linear motion blur kernels can be identified and estimated. The results will display the sampled sinogram which under reconstruction will yield the direct inversion estimate, the interpolated sinogram and the obtained *Sinogram Interpolated* (SI) kernel. This is done to illustrate the capability of the interpolation technique under real world circumstances.

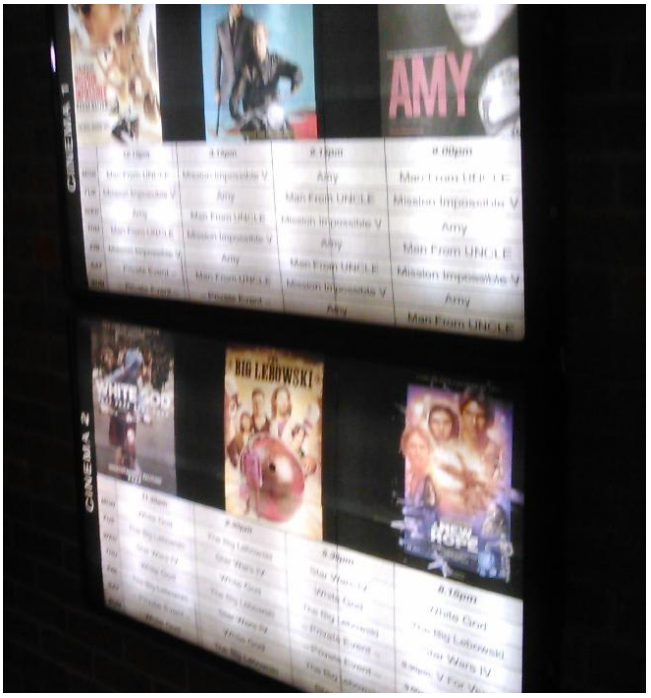


Figure 7.1: The blurred photograph.

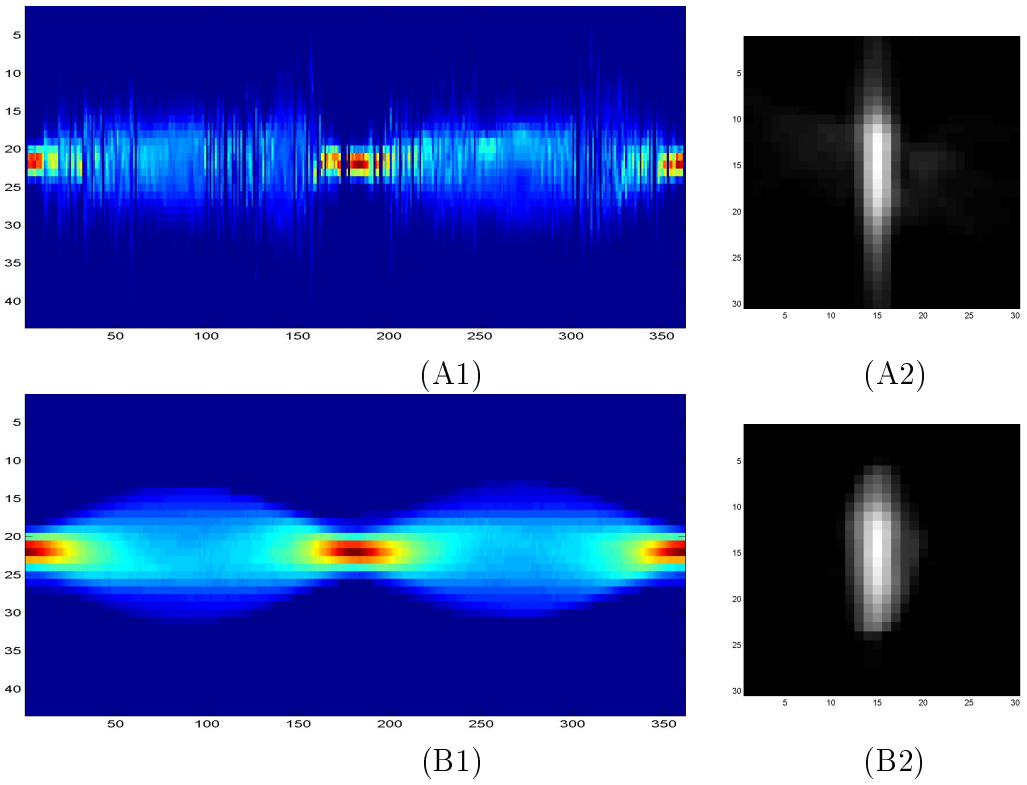


Figure 7.2: Sinograms and blur kernels, the sampled sinogram (A1) and resulting blur kernel (A2) and the interpolated sinogram (B1) and the resulting blur kernel (B2).

Figure 7.2 shows the kernels obtained from direct inversion of the sampled sinogram and the interpolated sinogram. It can be seen that the reconstruction artifacts that occur in Figure 7.2:(A2) are not present in the interpolated kernel (B2).

(A) (B) (C)

Figure 7.3: Zoomed in images, (A) the blurred image, (B) deblurred with the direct inversion blur kernel, (C) deblurred with the blur kernel obtained from interpolated sinogram.

By comparing Figure 7.3:(B) and 7.3:(C) it can be seen that the interpolated kernel performs better and produces a sharper image, after deblurring the original photograph 7.3:(A). The kernel that is obtained here is a linear motion blur kernel and it is therefore still left to show that non-linear blur kernels can be found.
The results that follow will not show the sampled sinogram and the obtained direct inversion kernel. This is done to make the analysis of the results more compact.

Result: Test photograph 2 In this case the photograph was affected by motion blur by non-linear motion blur. It will be shown that a non-linear motion blur kernel can be estimated and that a sharper deblurred photograph can be obtained by deconvolution with the estimated blur kernel.

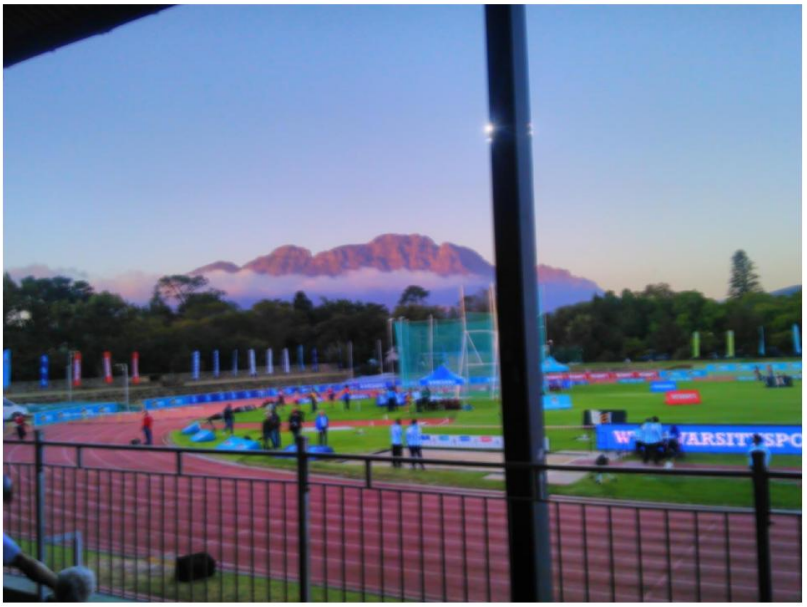
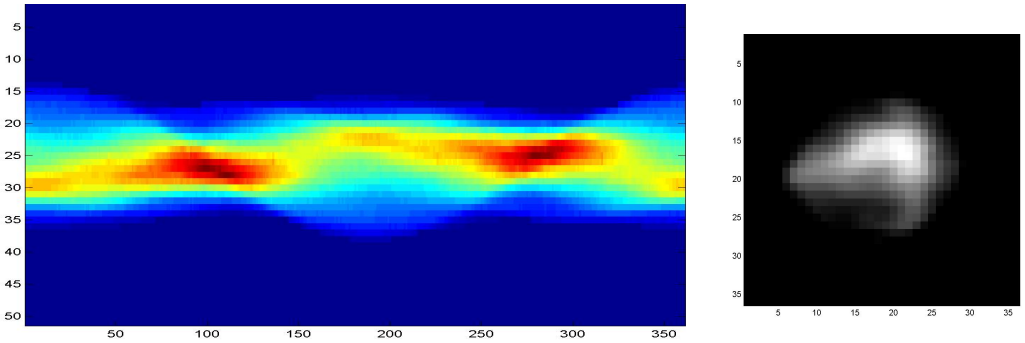
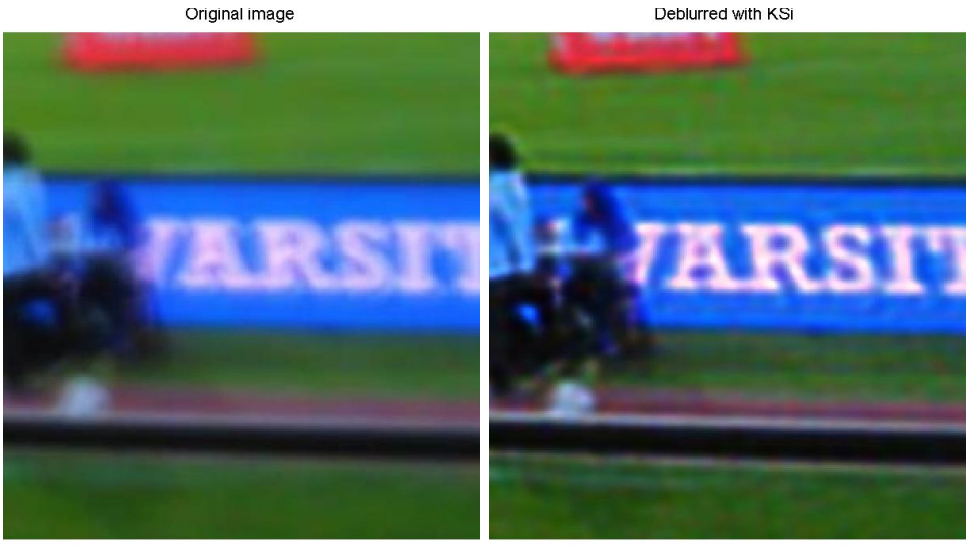


Figure 7.4: The blurred photograph.



(A1) (A2)

Figure 7.5: The interpolated sinogram (A1) and resulting blur kernel (A2).



(A) (B)

Figure 7.6: Zoomed in images, (A) the blurred image, (B) deblurred with the blur kernel obtained from interpolated sinogram.

Figures 7.4, 7.5, 7.6 display the estimated kernels and deblurred results for a photograph affected by a non-linear blur kernel. The results show that a non-linear blur kernels can be found. However, it does seem as though the obtained kernel (Figure 7.5:(A2)) is thicker than expected and as a result, the deblurred image (Figure7.6:(B)) contains some ringing artifacts as a result of this error.

Result: Test photograph 3

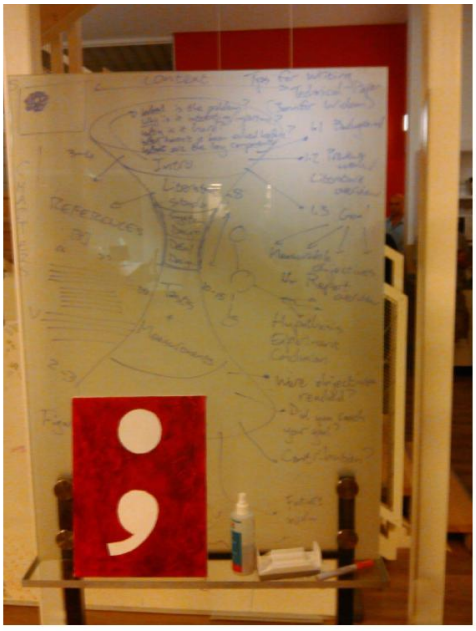
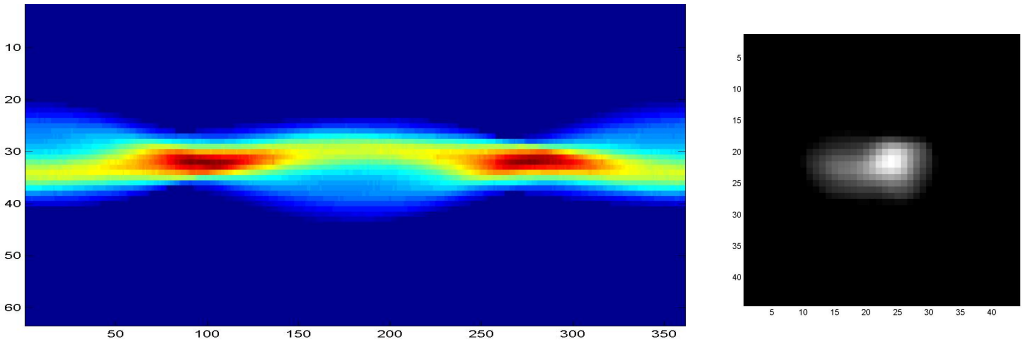
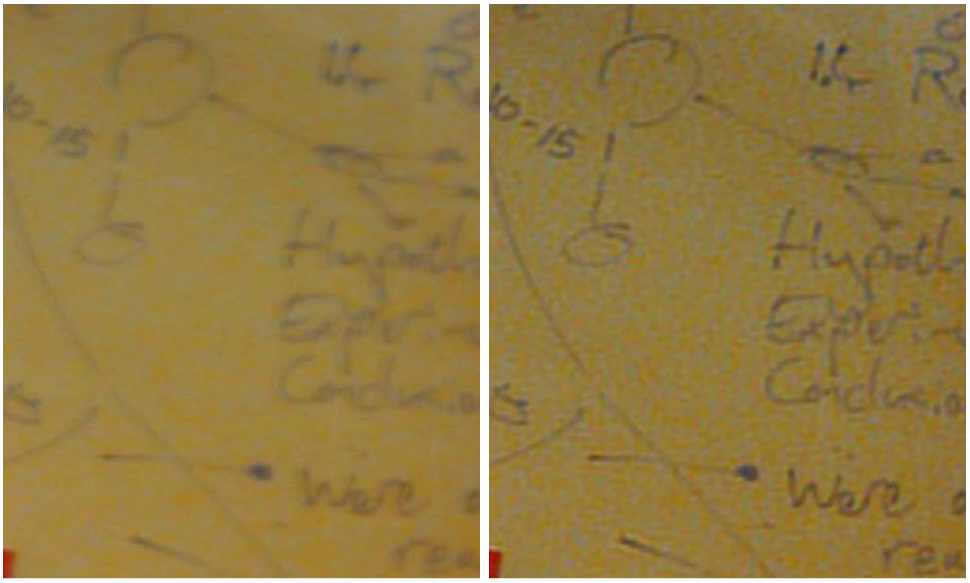


Figure 7.7: The blurred photograph.



(A1) (A2)

Figure 7.8: The interpolated sinogram (A1) and resulting blur kernel (A2).



(A) (B)

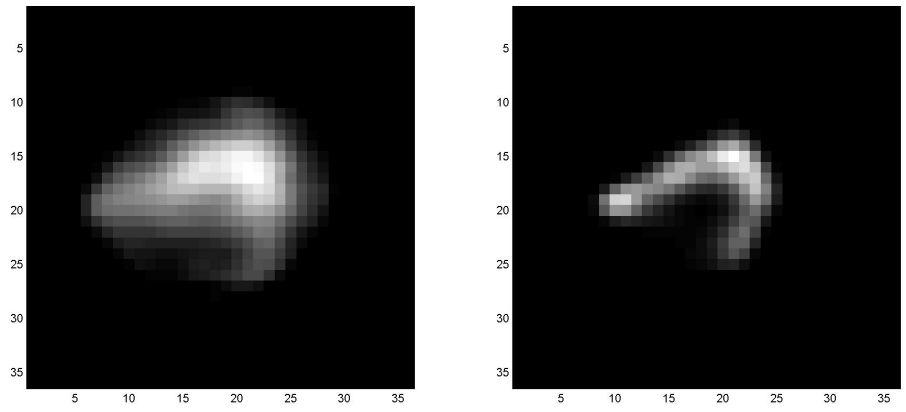
Figure 7.9: Zoomed in images, (A) the blurred image, (B) deblurred with the blur kernel obtained from interpolated sinogram.

These test photographs confirm that the tomographic estimation algorithm along with sinogram interpolation can be used to deblur motion blurred photographs.

7.1.2 Thinning the blur kernel

From the preceding results, it was noticed that the interpolated blur kernels tend to be a thick. The thickening of the kernel may be due to smoothing that is done while taking the derivative of the edge profiles or possibly slight over estimation of the contribution a valid sinogram warps has to a sinogram pixel when interpolating. We briefly show that the results obtained by thinning the blur kernel as a post processing step yields better deblurred images than those obtained from not thinning the kernel.It was decided that the interpolated blur kernels would thinned by deconvolution of the obtained blur kernel with a circular Gaussian function.

Result: Thinned improvement photograph 2



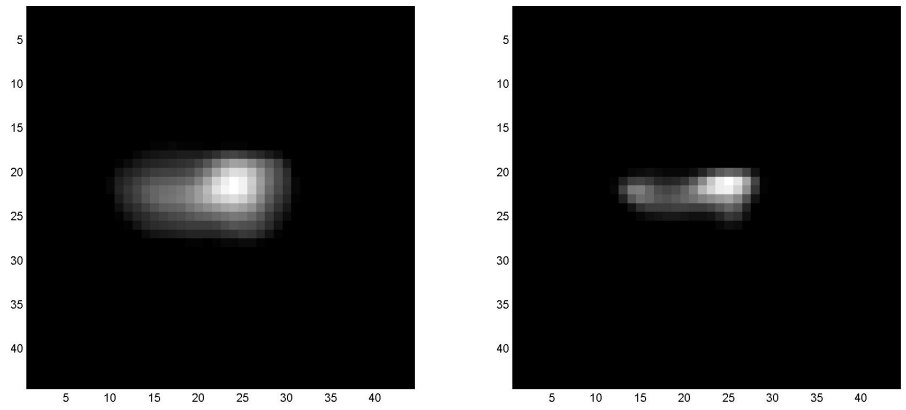
(A) (B)

Figure 7.10: Blur kernels, the interpolated blur kernel (A) and the thinned blur kernel (B).

(A) (B) (C)

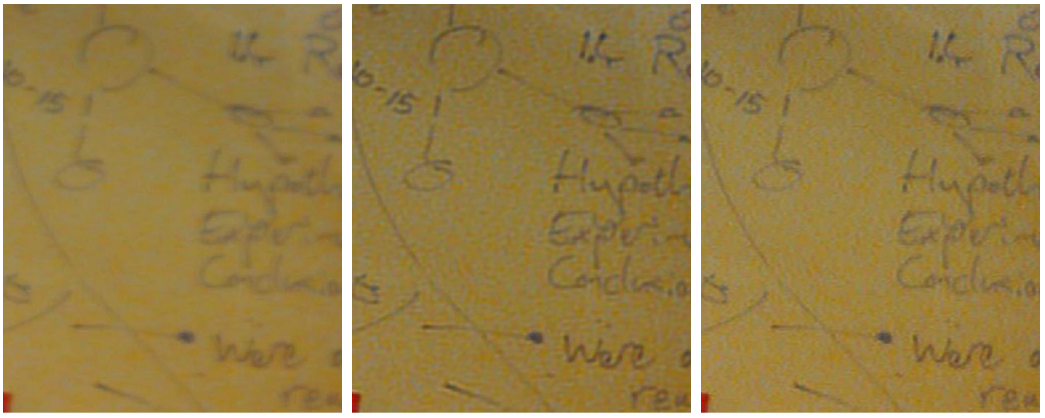
Figure 7.11: Zoomed image patches, the blurred image (A) deblurred with interpolated estimate (B), deblurred with thinned estimate(C).

Result: Thinned improvement photograph 3



(A) (B)

Figure 7.12: Blur kernels, the interpolated blur kernel (A) and the thinned blur kernel (B).



(A) (B) (C)

Figure 7.13: Zoomed image patches, the blurred image (A), deblurred using the interpolated estimate (B), deblurred image using the interpolated thinned kernel estimate (C).

It was found that deconvolution of larger kernels ($\texttt{size}(K) > 10 \times 10$) with a Gaussian filter with ($\sigma = 2 - 2.5$) produced the best thinned results. For smaller kernels ($\texttt{size}(K) < 10 \times 10$) it was found that ($\sigma = 1 - 1.5$) would provide better results. The thinned version of the topographically obtained kernel (Figure 7.12:(B2)), produce a better result than the non thinned version (Figure 7.12:(A2)). It can be seen from the deblurred images (Figure 7.13), that the thinned kernel produces fewer ringing artifacts and a sharper image.

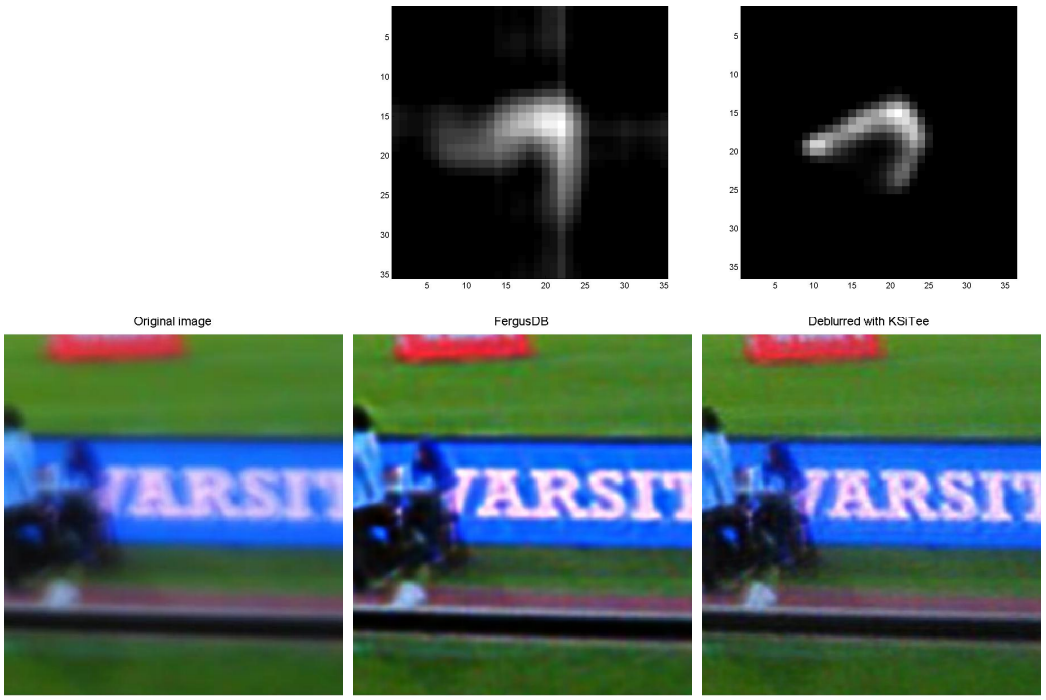
7.1.3 Comparison to *Fergus et al*

In this section we will try to draw comparison between the obtained SI kernels and those obtained using the algorithm described by Fegus *et al.* (2006). The results will attempt to show that the Radon method can produce similar results to those obtained by the natural image statistics technique of Fergus and possibly improve upon them. It was decided that only the thinned blur kernels would be compared the following section. The result obtained from the algorithm presented by Fegus *et al.* (2006) will be referred to as the Fergus Kernel.

Result: Comparison photograph 2

(A) (B)

Figure 7.14: Deblurred with Fergus kernel estimate (A), deblurred with tomographic estimation kernel (B).

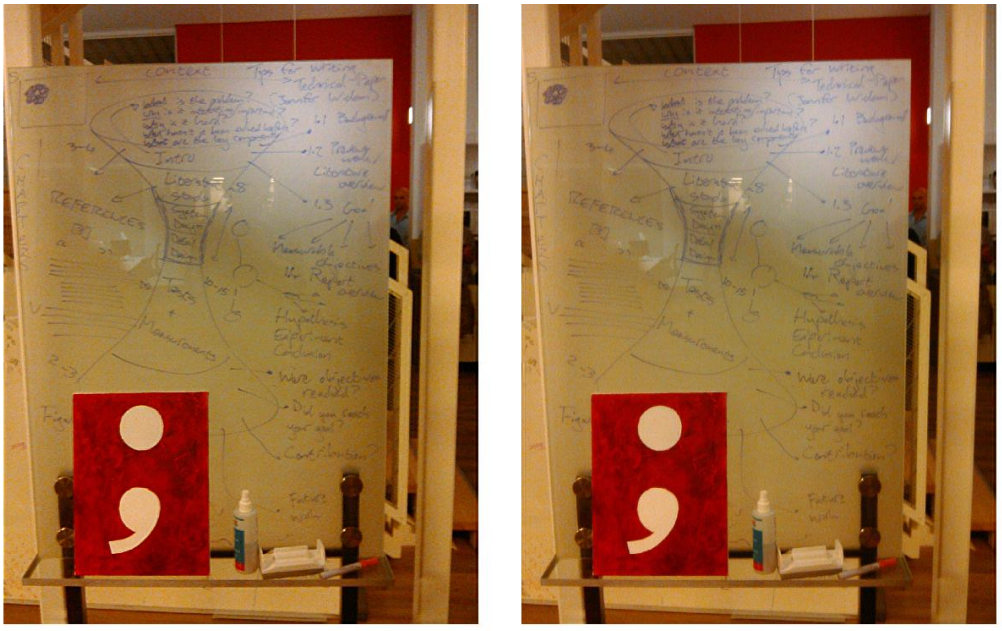


(A) (B) (C)

Figure 7.15: Zoomed in image patches, (A) the blurred image, (B) deblurred with Fergus kernel (C) deblurred with thinned tomographic estimation kernel.

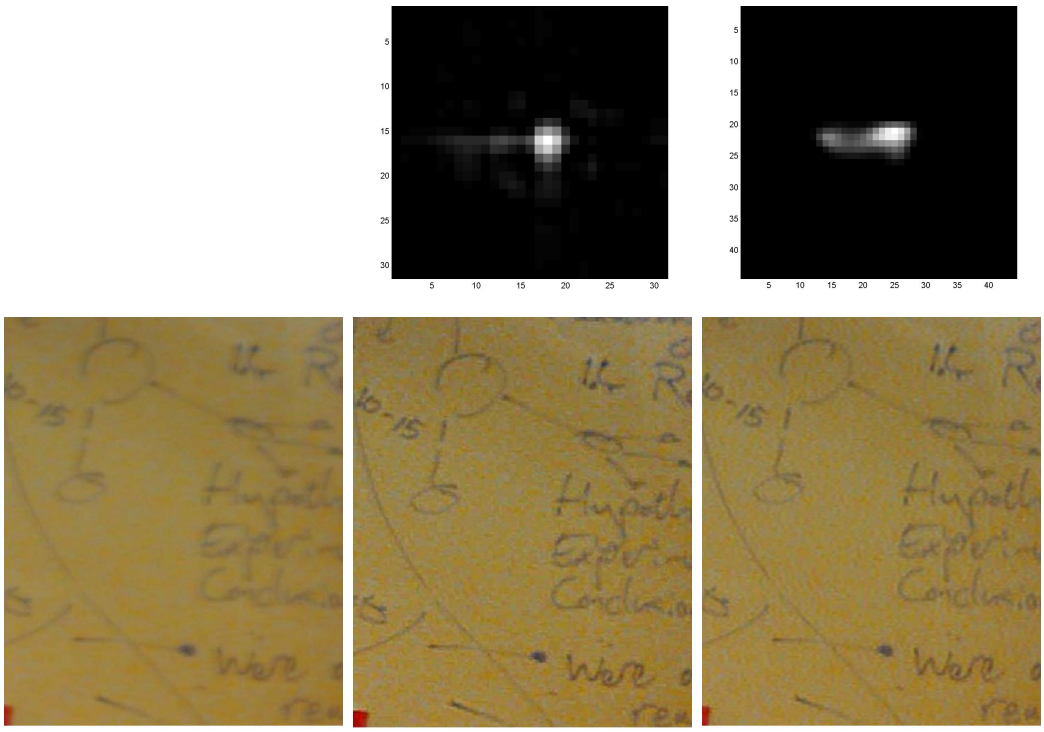
The obtained blur kernel (Figure 7.15;(C)) is similar to the one estimated by the algorithm described by *Fergus et al*. This shows that the Radon method can obtain comparable results. However the kernel obtained by *Fergus et al* does seem to produce a better deblurred result. Neglecting the kernel artifacts that can be seen slightly below and above the body of the kernel (Figure 7.15;(B)) estimated by *Fergus et al*, it can be gathered that the tomographic algorithm had not picked up the tail of the blur kernel which seems to trail away to the left of the body of the kernel. The kernel values in the tail of the kernel are small in comparison to those found close to the main body. This implies that the tomographic estimation did not pick up these values during the edge sampling steps or that these values were destroyed during by averaging noise reduction step, the sinogram interpolation or reconstruction steps.

Result: Comparison photograph 3



(A) (B)

Figure 7.16: Deblurred with Fergus kernel estimate (A), deblurred with tomographic estimation kernel (B).



(A) (B) (C)

Figure 7.17: Zoomed in image patches, (A) the blurred image, (B) deblurred with Fergus kernel, (C) deblurred with thinned tomographic estimation kernel.

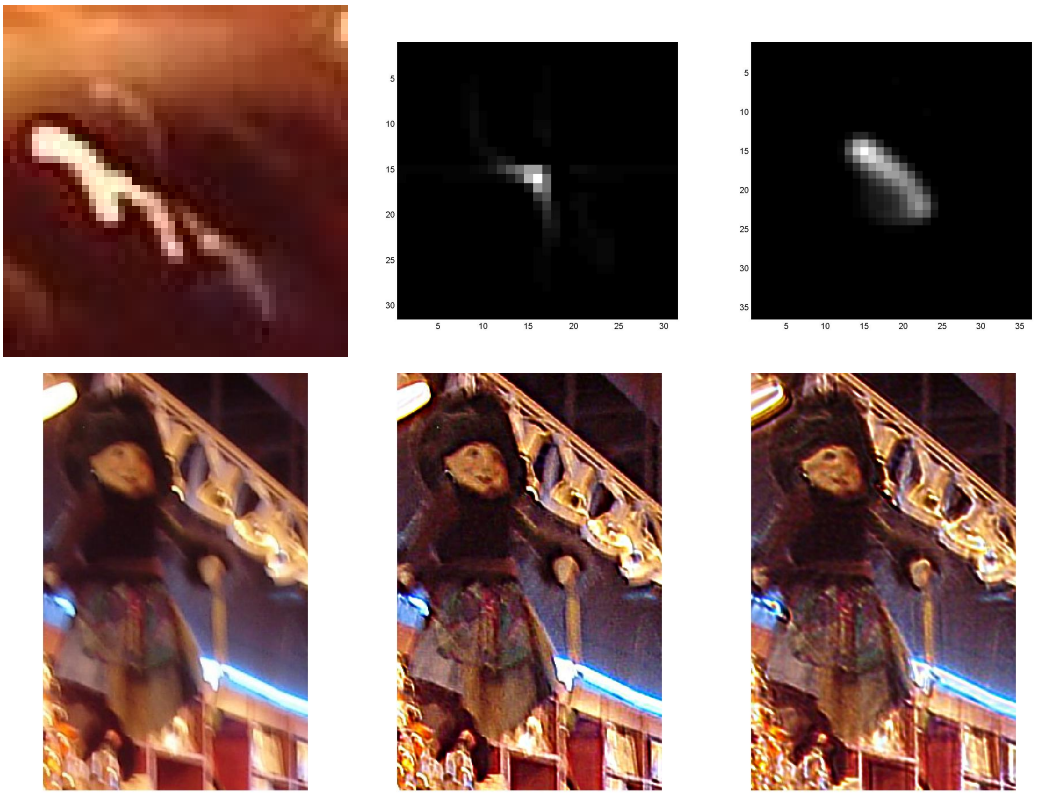
Here both estimation algorithms produce kernels with the similar right skewed horizontal motion, see Figure 7.17:(B) and (C). It is apparent from the estimates that the original blur kernel was indeed skewed. However, Figure 7.17:(B) does show the same slight vertical curvature in the middle of the kernel, that is seen in Figure 7.17:(C). It is difficult to judge the apparent quality of the obtained tomographic blur kernel, as both kernel estimates produce sharper deblurred image. Nonetheless, this result illustrates that the tomographic kernel can obtain an estimate for smaller blur kernels.

Result: Comparison photograph 4

Figure 7.18: The blurred photograph.

(A) (B)

Figure 7.19: Deblurred with Fergus kernel estimate (A), deblurred with tomographic estimation kernel (B).



(A) (B) (C)

Figure 7.20: Zoomed in image patches, (A)a blur artifact and the blurred image,(B) deblurred with Fergus kernel, (C) deblurred with thinned tomographic estimation kernel.

Figure 7.20 (B) and (C), shows that deconvolution with either estimated kernel is successful in deblurring the test image, see Figure 7.18. The tomographic estimated kernel, Figure 7.20:(C), does however seem to be better matched to the blur kernel artifacts that can be seen in the image. Comparing the image closely, it can be seen that the tomographic estimate has been more successful in deblurring the shelf edges that can be seen behind the puppet.

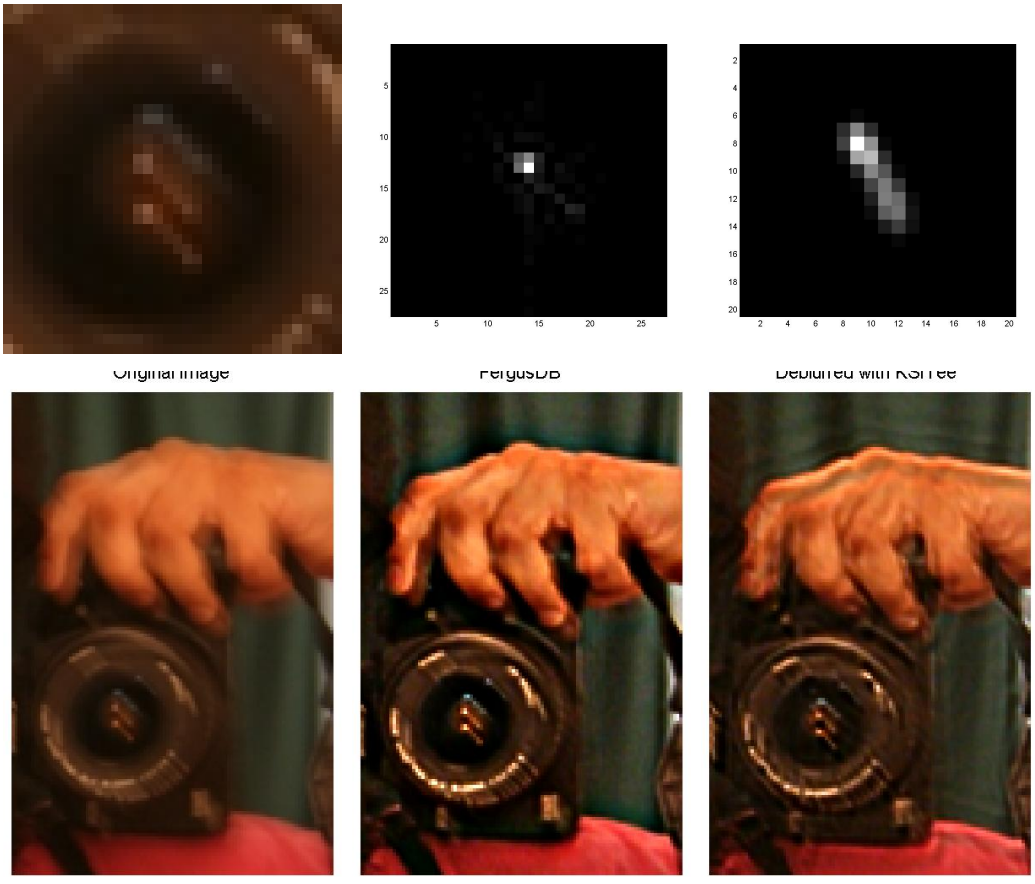
Result: Comparison photograph 5

The `mukta.jpg` test image is a featured test image in Fegus *et al.* (2006). Here we will compare the tomographic estimation result to the kernel that was obtained by the authors. The image and code are available online at Fegus *et al.* (2006).

Figure 7.21: The blurred photograph.

(A) (B)

Figure 7.22: Deblurred with Fergus kernel estimate (A), deblurred with tomographic estimation kernel (B).



(A) (B) (C)

Figure 7.23: Zoomed in image patches, (A) a blur artifact the blurred image, (B) deblurred with Fergus kernel, (C) deblurred with thinned tomographic estimation kernel.

The deblurred image (Figure 7.22:(B)), shows that the tomographic estimate result is able to deblur the `mukta.jpg` test image. However assuming that the optimized result found by Fergus is closer to the correct blur kernel [2] than the tomographic estimate. The tomographic estimated kernel is not as tapered as the fergus estimate because of the low number of edges available in the `mukta.jpg` test image. The image was however able to provide the tomographic algorithm with enough information to point out the general direction of the blur kernel, which allows for some deblurring to occur.

Result:Simulated Comparison photograph 1

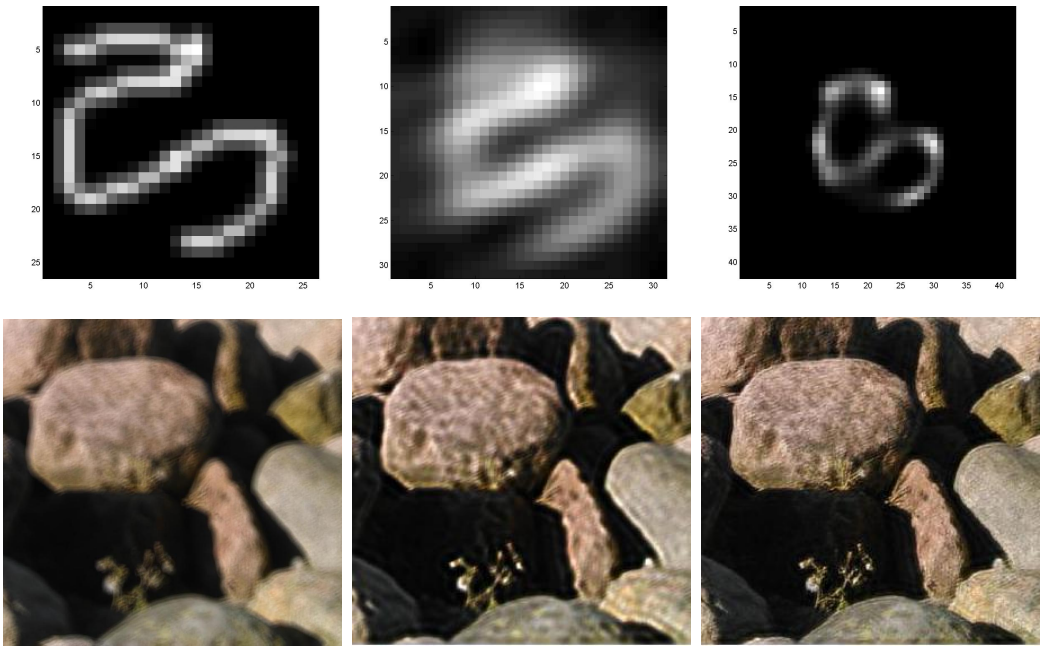
[2] A good estimate is the blur artifacts produced by point sources in the image, for the `mukta.jpg` test image such artifacts can be seen on the lens of the camera.

Here the estimated kernel results will be compared to a known true blur kernel that was simulated.

Figure 7.24: The blurred photograph.

(A) (B)

Figure 7.25: Deblurred with Fergus kernel estimate (A), deblurred with tomographic estimation kernel (B).



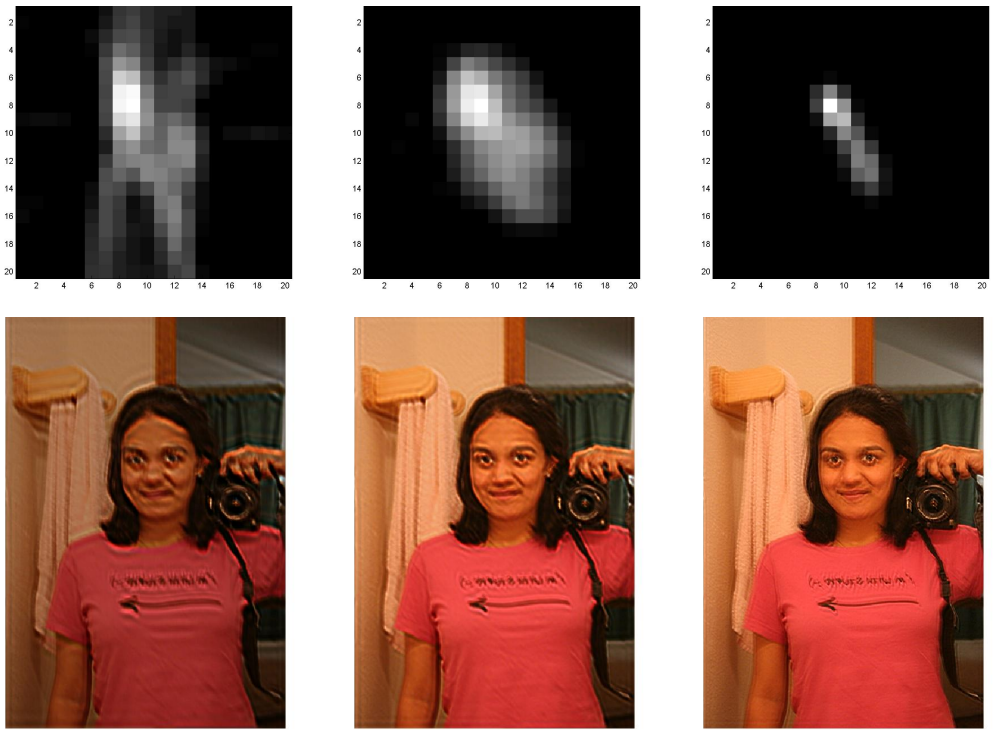
(A) (B) (C)

Figure 7.26: Zoomed in image patches, (A) the blurred image, (B) deblurred with Fergus kernel, (C) deblurred with thinned tomographic estimation kernel.

This result shows that the tomographic algorithm can deliver a comparable result even when dealing with extreme motion blur.

7.1.4 Discussion

The various results show that the tomographic estimation algorithm can be used to estimate linear and non-linear blur kernels for motion blurred photographs. However, as expected the performance of the tomographic algorithm depends on the number of edge profiles that can be sampled from the blurred image. In all the results shown the algorithm was able to estimate the blur kernel with varying accuracy.
The comparison to the kernels found by *Fergus et al*, confirms that the tomographic estimate can deliver comparable results in various cases (see Figures 7.15,7.17) and possibly improve on in the kernels that are found by *Fergus*, see Figure 7.20. The results also highlights a point that needs to be addressed in order for the tomographic algorithm produce blur kernels more accurately, namely, to thin the obtained interpolated kernel sinograms which have been smoothed. It is clear that the thinned version of an interpolated sinogram produces a better kernel results and therefore better deblurred images, see Figure 7.27.



(A) (B) (C)

Figure 7.27: Blur kernels and their deblurred images respectively: (A) Deblurred with directly inverted sample kernel. (B) Deblurred with interpolated kernel. (C) Deblurred with thinned interpolated kernel.

These results show that MAP estimation can be avoided even when dealing with motion blurred images. The algorithm developed by Cho *et al.* (2011) utilizes sparse priors on the blur kernel to estimate the kernel from the obtained sinogram data, using a MAP estimation technique. As explained in chapter 2, this boils down to minimizing the negative log likelihood over all possible blur kernels. An initial blur kernel K^1 is estimated from assumed priors and the sinogram $S_{K^1}(\theta, h) = \mathfrak{R}[K^1(x, y)]$ of the kernel is obtained. This estimate is refined by minimizing the error between $S_{K^t}(\theta, h)$ and the sinogram $S_B(\theta, h)$ found by sampling edges in the blurred image B, that is

$$K \approx \min_{K^t}\{||S_{K^t}(\theta, h) - S_B(\theta, h)||^2\}.$$

This places it in the algorithms similar to Joshi *et al.* (2008) and Fegus *et al.* (2006)[3] and therefore has the same disadvantages.

[3]The algorithms described by Cho *et al.* (2011), Joshi *et al.* (2008) and Fegus *et al.* (2006) use data obtained from the blurred image and estimate the kernel from this data using a MAP technique.

A minimum search is not guaranteed to find the global minimum [4] that resides within the set of parameters. Such searches are also dependent on assumptions [5] which are made in order to define the search parameters. Parametrization of the blur kernel also limits the estimation procedure to a distribution subspace, which excludes real world variations that may be found in photograph blur kernels.

Given this, it is important to realize that the tomographic estimation algorithm, up to this point, is a direct method. It directly inverts interpolated image data to reconstruct the blur kernel. It has been demonstrated that the tomographic algorithm in combination with sinogram interpolation and thinning is capable of estimating non-linear blur kernels, comparable to the success of the Fegus *et al.* (2006) algorithm. It is hoped that with further improvement, the tomographic algorithm will best the results produced by Fegus *et al.* (2006) especially when dealing with extreme non-linear blur.

[4] This is also pointed in Fegus *et al.* (2006).

[5] It is often assumed the kernel is a smooth Gaussian mixture.

Chapter 8

Conclusion and future work

8.1 Future work and notes on implementation

Note on the algorithm

During the testing it was noted that the angle bins can often contain a large number of edges, especially when dealing with large images with a great number of edges. It must therefore be considered if the choice of having a fixed bin range, of one degree for the edge angles is optimal. Although testing was done with larger intervals it was found that the one degree bin range and step produced the best results. Given the relative simplicity of this problem it seemed that the best method would be to employ an algorithm such as k-means to optimize the bin locations and intervals.

The effect of a blur kernel on image values is reduced within regions of constant or close to constant value. Similarly when an edge is blurred, the effect of the blur kernel is less prominent when the edge minimum and maximum are close together, i.e. when a_0, a_1 are close together, see equation (4.2.2). Because digital images are stored as matrices with values between 0 and 255 edge profiles where the minimum and maximum are far apart, will have a wider range (integer values between $50 - 200$) of possible blurred edge values than edges where the minimum and maximum are close together(integer values between $25 - 75$). Consequently edge profiles with a wide minimum-maximum range will contain better kernel projection data and produce better edge derivatives. A wider minimum-maximum range is also less susceptible to image noise.

The use of scaled edges with too small a minimum-maximum range may be the reason for the destruction of kernel values which are small in comparison to the maximum kernel value, see Figure 7.15:(B),(C). These edge profile val-

ues are dealt with during the profile averaging step see Chapter 5.9.3. During the averaging step outliers are removed and the scaled edge profiles are combined to produce a mean edge profile for each angle bin. Given the effect the minimum-maximum range can have on the kernel value it must be considered wether it would be possible to produce a weighted average profile to replace the mean scaled profile. A weighted average method may also improve the removal of outliers.

After having completed the averaging step it must be considered whether the sampled edge profiles must be derived directly using discrete derivatives or whether it would be better to model the edge as a continuous function, such as a polynomial, produce the edge derivative from the fitted model of the edge.

It has been shown that the sinogram interpolation method that is used to improve the sampled kernel sinogram does in fact improve the obtained kernel and is vital to produce accurate blur kernels. However the big question that must be asked, is if the described technique for improving the sinogram is indeed the most optimal sinogram interpolation method for blur kernels. Many sinogram interpolation methods are focussed on the interpolation of regularly spaced and non sparse sinograms. The technique described in this thesis, for the use of interpolating blur kernel sinograms, most importantly can interpolate sparse random sinograms as most images do not contain edges at regularly spaced orientations. However, our interpolation technique is not the only sparse and random interpolation technique that exists. A tests of this nature to see if other interpolation methods may provide better results is beyond the scope of this thesis.

On the matter of improving the described sinogram interpolation technique various steps can be considered. Firstly, the number of warps that are found can be increased by decreasing the step size between original pixel positions. This would essentially allow for a higher interpolation resolution. It would also be beneficial to improve the accuracy of the warp values that are found.

Thinning

Further work also needs to be done to tackle the over estimation and thickening of the obtained blur kernels. Throughout testing a two dimensional Gaussian function was used to thin the interpolated sinogram columns, using deconvolution, as this provided better results than the use of an averaging function. Standard deviation (STD) values of 2 with a matrix size of 11×11, yielded the best thinning results. It is difficult to determine a single STD value that should be used for all estimated and interpolated sinograms, as the thickening of the kernels is also affected by the number of good edges that are found, the size of the blur kernels and the sample step size. Work on this area is duely needed to

track down the cause and eliminate the thickening of the blur kernels correctly.

Including lines and other image features

Throughout this thesis edges were used to capture the Radon transform of the blur kernel. It is however possible to track down the Radon transform in other image features as well. A great example of this is lines which will directly provide the profile from sampling the blurred line and scaling it. We refer the reader to Appendix A in which the Radon transform theory for blurred lines is developed and illustrated. The mathematics and the illustration shows that lines can be used to obtain the Radon transform of the blur kernel. We believe that by following a similar approach as done in appendix A and with the ideal edge case, see section 4.2, that the Radon theory can be expanded to encompass not only blurred lines and edges but specifically corners and other traceable image features.

Improving sampled sinograms

It would be beneficial if a method could be developed to check that obtained sinogram columns are correlated and not overly affected by noise or sampling error. This would essentially be a check of the "sinogramicity" of the obtained columns in relation to each other to ensure that only the best sinogram columns are used in the sampled blur kernel sinogram as well as the interpolation step. This will essentially remove outlier columns and therefore produce a better interpolated sinogram with less noise.
We suggest that such a method can be based on the RANSAC algorithm. The set of warp curves that are found for a random number of sinogram columns would then form the model that is obtained in each RANSAC iteration.

Utilizing the extended blur kernel model

Once the kernel estimate has been found, it would be use fully to reconstruct the blur kernel according to the extended blur model, that is

$$K(x, y) = A(x, y) * (S(x, y) * P(x, y)) + N(x, y),$$

where $S(x, y)$ is the kernel motion path and $P(x, y)$ the basis kernel. This would purely involve identifying the kernel motion path and basis kernel. It is suggested that morphological treatment of the obtained interpolated blur kernel could be used to find the kernel path and intensities, while the basis kernel could be identified by identifying the parameters of the smallest recurring peak within the sampled sinogram.

Improving run times

Although no though run time testing could be done on the algorithm it is important to note some important aspects which may speed up the computation of blur kernels with the Radon transform and sinogram interpolation methods. Similar to many image processing algorithms the process would be sped up considerably by pre identifying areas of interest. In our case this would be identifying image regions that contain good edges. Secondly as a whole the algorithm has the possibility to utilize parallel computing methods. One example may be to run the RANSAC step in parallel with the edge sampling.

8.2 Conclusion

The tomographic algorithm which was developed and implemented in this thesis has proven to be a viable and accurate method for finding motion blur kernels. It has also been shown that these blur kernels can then successfully be used to deblur blurred photographs. However this can only be done if the photograph contains a sufficient number of good edges spanning various angles. Having encountered the reconstruction artifact problem for sparse random sinograms, a sinogram interpolation technique was developed to further improve the kernels that are obtained. It was shown that by the inclusion of the sinogram interpolation technique, kernel estimation could be done to a greater accuracy and that estimates could still be produced for images with a lower number of good edge profiles than previously[1]. The blur kernels that were obtained were then improved by thinning and shown to be comparable to other modern motion blur kernel estimation algorithms.
It is important to note method described by this thesis obtains the kernels based on only three assumptions that are made about the blurred image and blur kernel. These were:

- The photographs and images that are dealt with, are all translation motion or aperture blurred. Blur occurring from camera rotation is not considered.
- The blur for all images and photographs is spatially invariant. That is, the whole image was affected by the same blur kernel and motion.
- The blur kernel is positive and energy conserving, that is,

$$1 = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} K(x, y) dx\, dy.$$

Further no smoothness, noise, or distribution assumptions are made about the blur kernel and the obtained kernels are therefore non-parametric. We were

[1] If perfect sinogram data can be obtained from an image this could be as low as 10 sinogram columns.

able to show that a successful blur kernel estimation algorithm can be created with these small number of assumption that were made.

Although work needs to be done to further improve the results, the obtained blur kernels already show great promise. It can therefore finally be concluded that the estimation of non-parametric motion and aperture blur kernels, by utilizing a direct estimation algorithm, can indeed be done by use of image sampling, the Radon transform and sinogram interpolation. Lastly, the obtained blur kernels can then be used successfully by known deconvolution methods to deblur and restore the blurred photographs.

Appendices

Appendix A

Including lines

Here it will be shown that the radial profile of the blur kernel can also be obtained from lines within a blurred image. Consider a single line in an image A_θ, where

$$A_\theta(r,t) = \begin{cases} a_0, & \delta(r - r'\cos\theta) \\ a_1, & \text{otherwise} \end{cases}$$

where $(x,y) = (r\cos\theta - t\sin\theta, t\cos\theta + r\sin\theta)$

$$B_\theta(r,t) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} A_\theta(r',t')K_\theta(r-r',t-t')dr'\,dt'. \tag{A.0.1}$$

When $t = 0$,

$$B_\theta(r,0) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} A_\theta(r',t')K_\theta(r-r',-t')dr'\,dt',$$

$$= \int_{-\infty}^{\infty} \lim_{n\to 0}\left[\int_{-\infty}^{r=-n} a_1 K_\theta(r-r',-t')dr' + \int_{r=-n}^{r=n} a_0\delta(r-r'\cos\theta)K_\theta(r-r',-t')dr' + \int_{r=n}^{\infty} a_1 K_\theta(r-r',-t')dr'\right]dt'$$

$$= \int_{-\infty}^{\infty} \lim_{n\to 0}\left[\int_{-\infty}^{r=-n} a_1 K_\theta(r-r',-t')dr' + \int_{r=n}^{\infty} a_1 K_\theta(r-r',-t')dr' + a_0 K_\theta(r,-t')\right]dt'$$

$$= a_1\int_{-\infty}^{\infty} \lim_{n\to 0}\left[\int_{-\infty}^{r=-n} K_\theta(r-r',-t')dr' + \int_{r=n}^{\infty} K_\theta(r-r',-t')dr'\right]dt' + a_0\int_{-\infty}^{\infty} K_\theta(r,-t')dt'$$

$$= a_1\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} K_\theta(r-r',-t')dr'dt' + a_0\int_{-\infty}^{\infty} K_\theta(r,-t')dt'$$

Therefore we have that,

$$B_\theta(r,0) = a_1 + a_0\int_{-\infty}^{\infty} K_\theta(r,-t')dt' \tag{A.0.2}$$

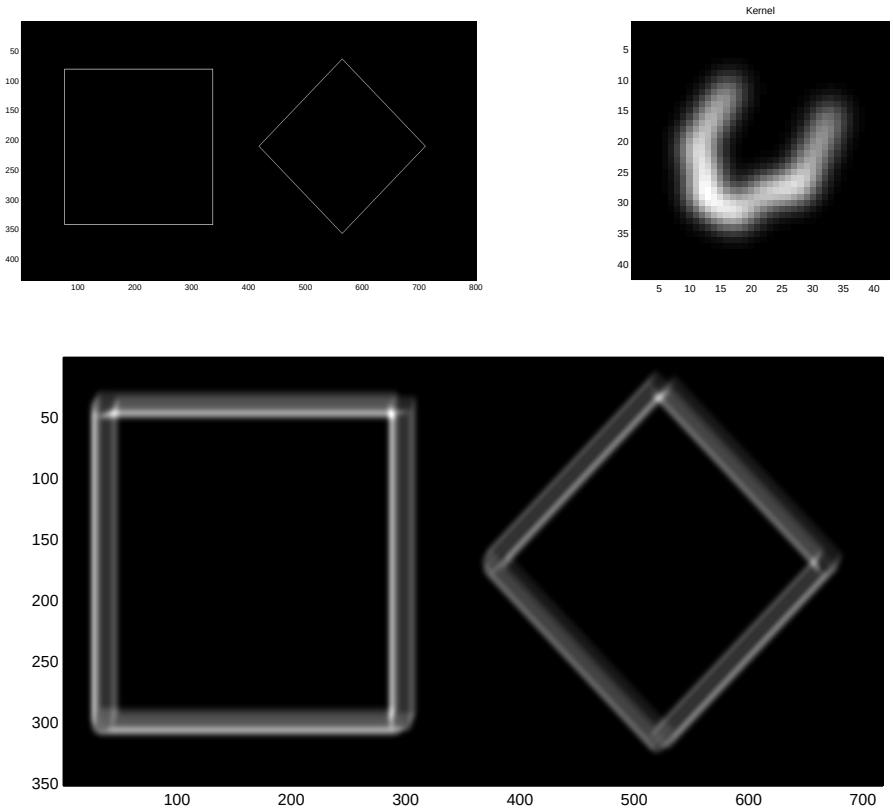


Figure A.1: The test image containing lines, the simulated blur kernel and the blurred test image.

Rearranging equation A.0.2 it is shown that,

$$\frac{B_\theta(r,0) - a_1}{a_0} = \int_{-\infty}^{\infty} K_\theta(r, -t')dt'. \tag{A.0.3}$$

This implies that by simply sampling a line in a blurred image $B(x, y)$ the Radon transform of the blur kernel $K(x, y)$ at angle θ can be retrieved.

The appearance of the Radon transform of the kernel across lines in an image is easily illustrated by blurring a test image with a blur kernel, Figure A.1. Comparing the kernel sinogram with the blurred lines in the image, Figure A.2, it can be seen that the blurred lines in the image are almost an exact match for the four sinogram columns.

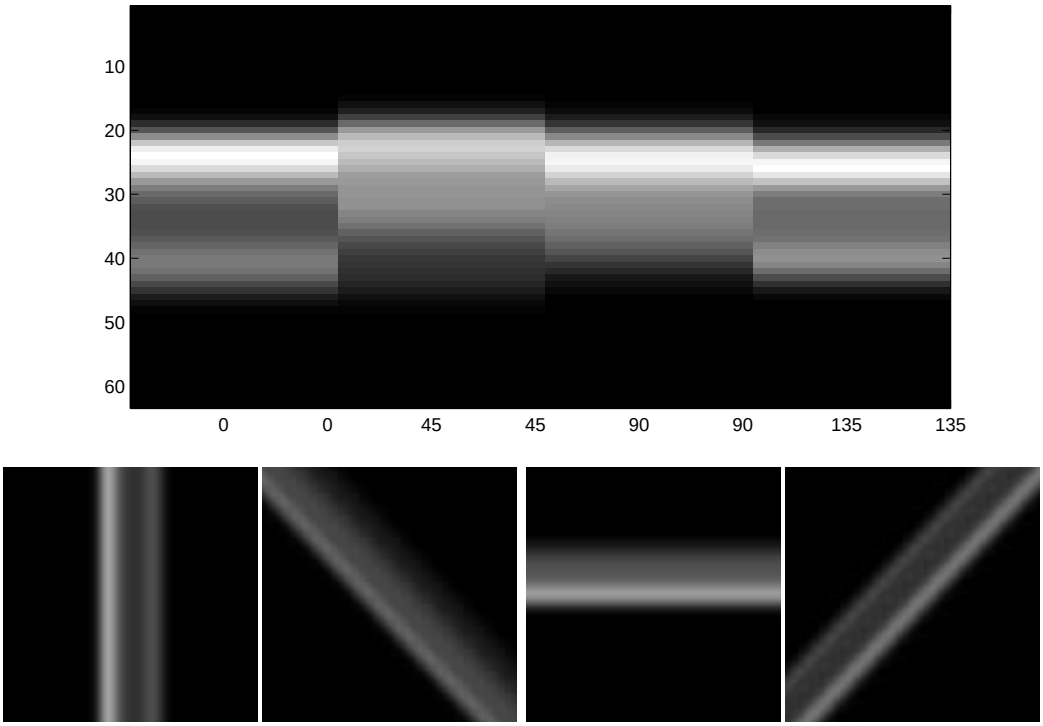
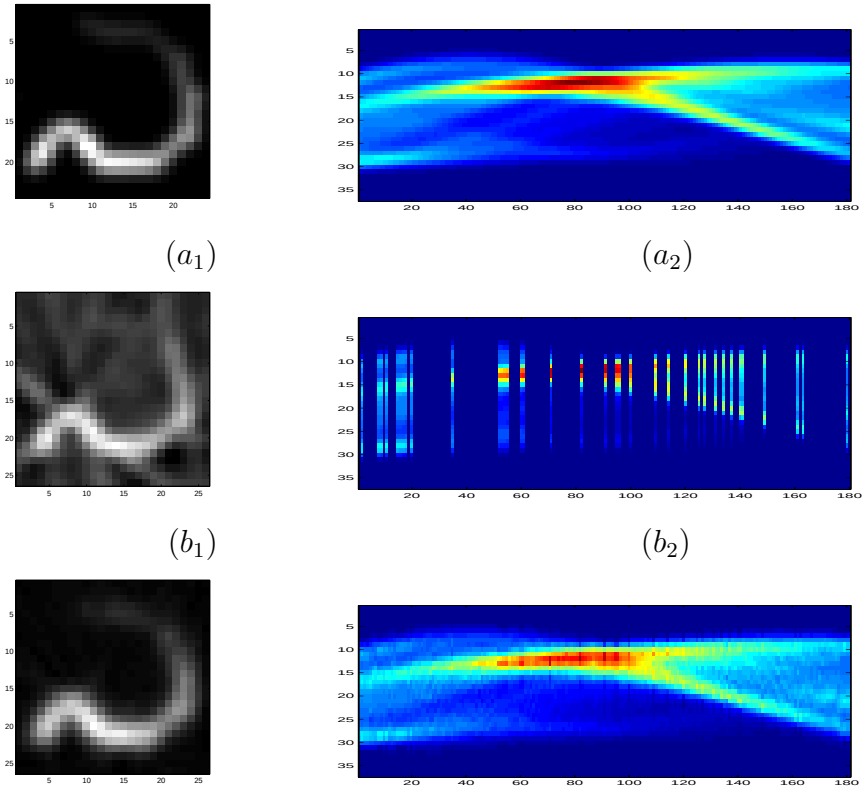


Figure A.2: The blur kernel sinogram with columns at $\theta = 0^o, 45^o, 90^o, 135^o$ and zoomed in versions of the blurred lines from the test image.

Appendix B

Interpolation results

Simulated blur kernel results :



(a_1) (a_2)

(b_1) (b_2)

(c_1) (c_2)

Figure B.1: Forty sparse random sinogram columns test kernel B. The simulated blur kernel and its sinogram (a_1, a_2)), the sparse sinogram and the recovered kernel(b_1, b_2), the interpolated sinogram and recovered kernel(c_1, c_2).

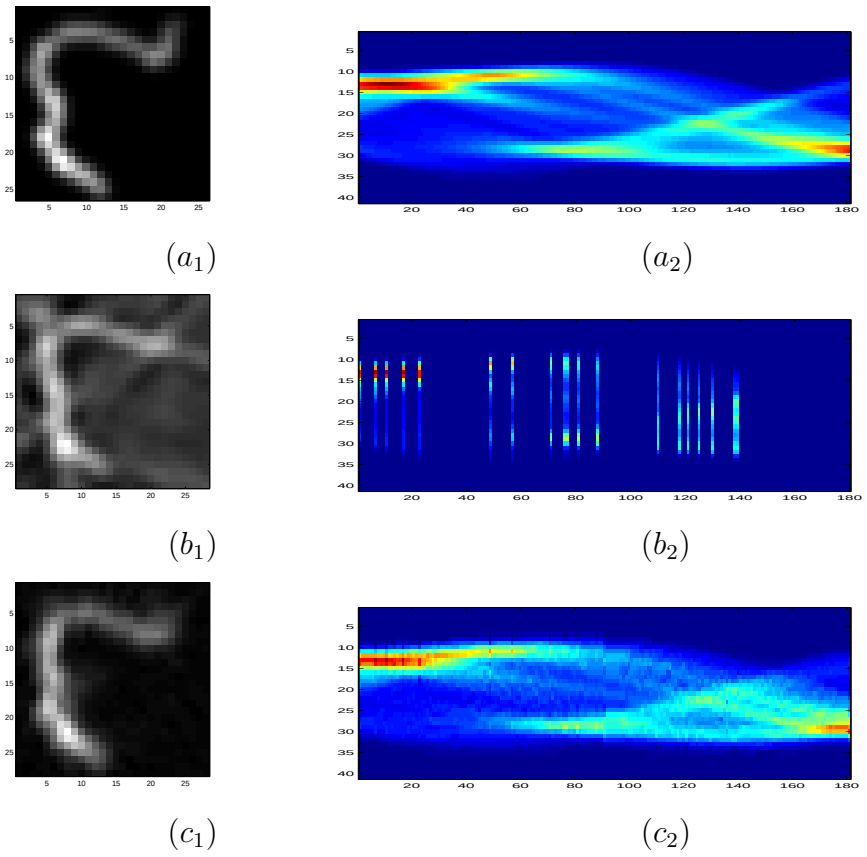


Figure B.2: Twenty sparse random sinogram columns test kernel B. The simulated blur kernel and its sinogram (a_1, a_2), the sparse sinogram and the recovered kernel(b_1, b_2), the interpolated sinogram and recovered kernel(c_1, c_2).

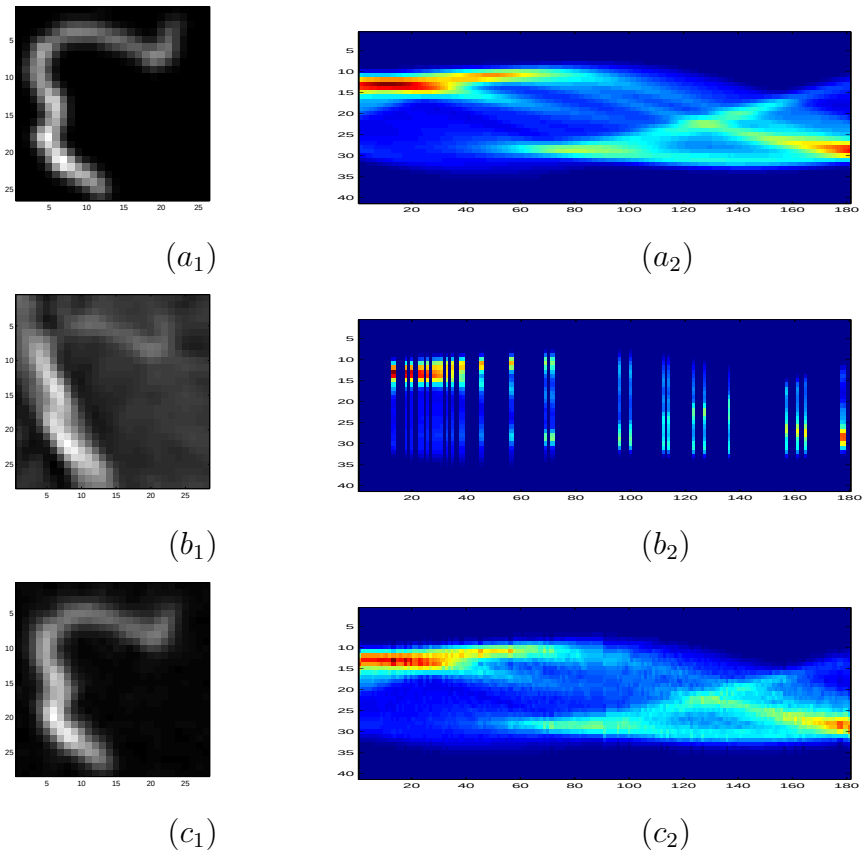


Figure B.3: Forty sparse random sinogram columns test kernel B. The simulated blur kernel and its sinogram (a_1, a_2), the sparse sinogram and the recovered kernel(b_1, b_2), the interpolated sinogram and recovered kernel(c_1, c_2).

Simulated blurred image results :

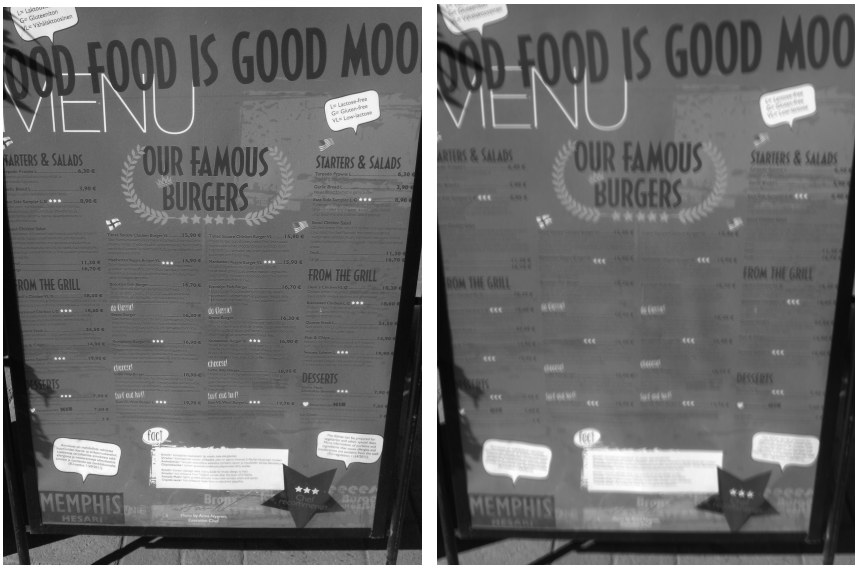


Figure B.4: The original test image and blurred test image

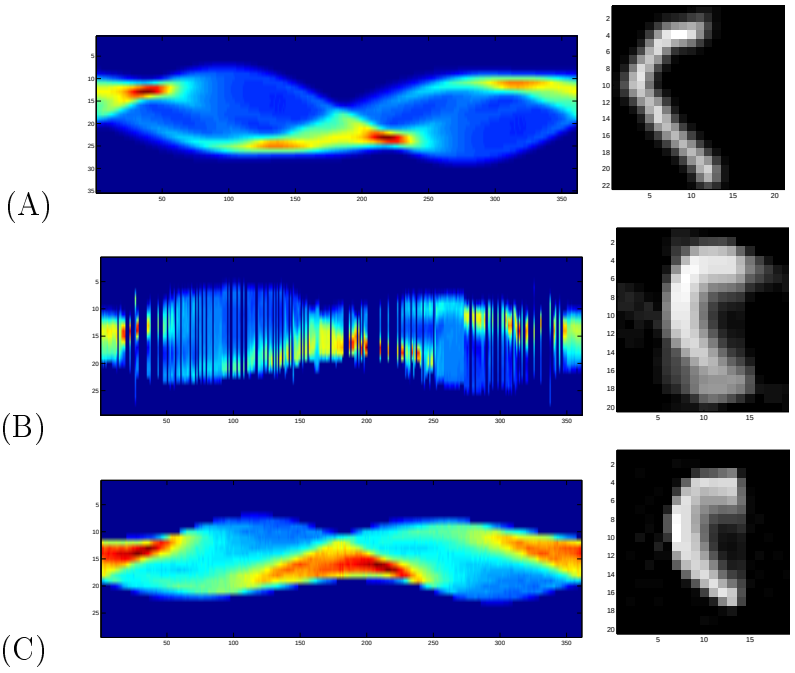


Figure B.5: Sinograms and blur kernels, (A) the original blur kernel and its sinogram, (B) the sampled sinogram and resulting blur kernel and (C) the interpolated sinogram and the resulting blur kernel.

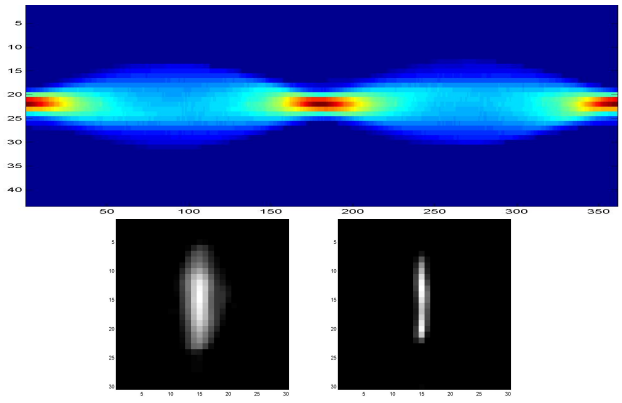
Figure B.6: Zoomed image regions, (A) the blurred image, (B) deblurred using the original blur kernel, (C) deblurred with the direct inversion blur kernel, (D) deblurred with the blur kernel obtained form interpolated sinogram.

Appendix C

Notes on results

This appendix presents further results that were collected from test of the tomographic estimation algorithm. These results are supplementary to the results displayed on pages 82 – 102. We further include the user inputs that were used in the accompanying code with which the various results were obtained.

Result: Test photograph 1
`Thesis code inputs:` `Kernelsize`= 20, `Samplestepsize`= 0.5, `ThetaBINsize`= 1, `thinsigma`= 2;



(B1) (B2)

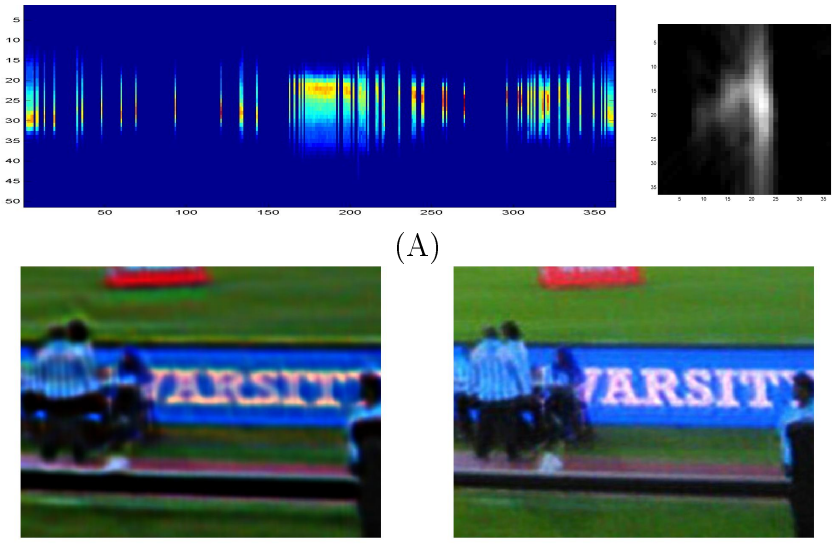
Figure C.1: Sinograms and blur kernels,(A) and the interpolated sinogram, (B1) the resulting kernel and the thinned blur kernel (B2).

(A) (B) (C)

Figure C.2: Zoomed in images, (A) the blurred image, (B) deblurred with the interpolated blur kernel, (C) deblurred with the thinned blur kernel.

Result: Test photograph 2

`Thesis code inputs: Kernelsize`= 25, `Samplestepsize`= 0.5, `ThetaBINsize`= 1, `thinsigma`= 2;



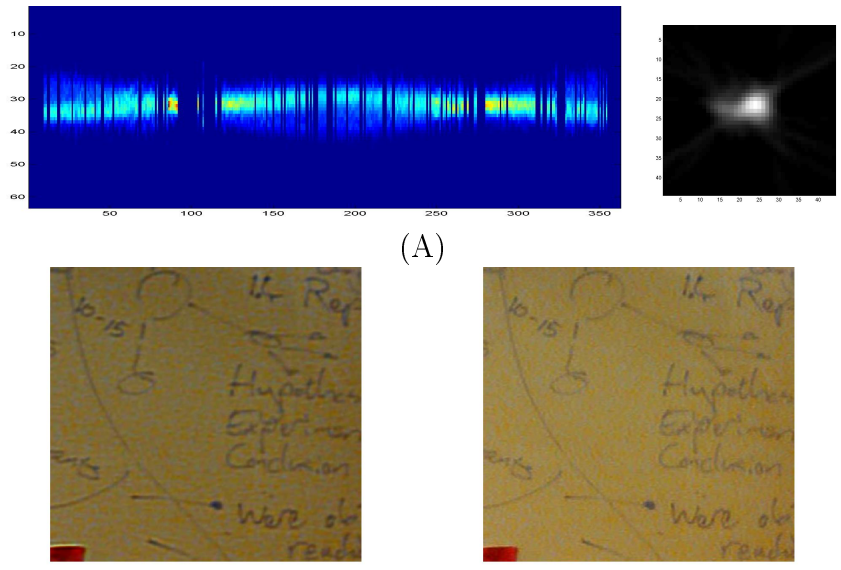
(A)

(B) (C)

Figure C.3: (A) Sampled sinogram and the resulting direct inversion kernel. Zoomed image region (B)of the deblurred image using (A), (C) deblurred using the interpolated thin version.

Result: Test photograph 3

`Thesis code inputs: Kernelsize`= 31, `Samplestepsize`= 1, `ThetaBINsize`= 1, `thinsigma`= 1.5;



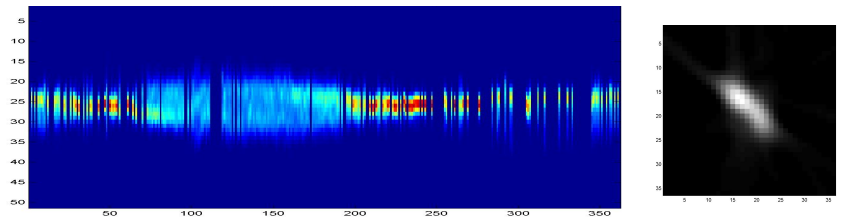
(A)

(B) (C)

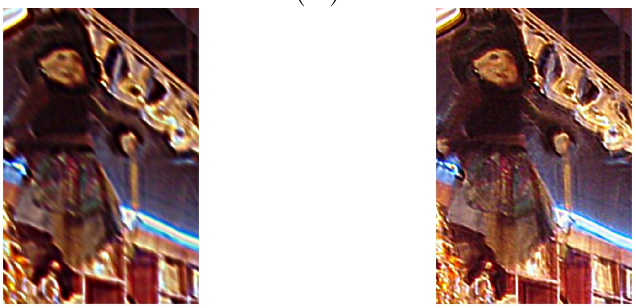
Figure C.4: (A) Sampled sinogram and the resulting direct inversion kernel. Zoomed image region (B)of the deblurred image using (A), (C) deblurred using the interpolated thin version.

Result: Test photograph 4

`Thesis code inputs: Kernelsize`= 25, `Samplestepsize`= 1, `ThetaBINsize`= 1, `thinsigma`= 2;



(A)

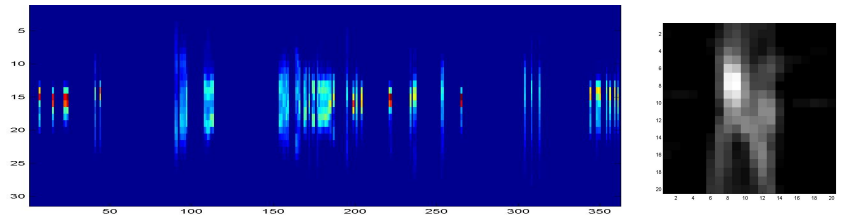


(B) (C)

Figure C.5: (A) Sampled sinogram and the resulting direct inversion kernel. Zoomed image region (B)of the deblurred image using (A), (C) deblurred using the interpolated thin version.

Result: Test photograph 5

`Thesis code inputs: Kernelsize`= 15, `Samplestepsize`= 0.5, `ThetaBINsize`= 1, `thinsigma`= 2;



(A)

(B) (C)

Figure C.6: (A) Sampled sinogram and the resulting direct inversion kernel. Zoomed image region (B)of the deblurred image using (A), (C) deblurred using the interpolated thin version.

List of References

Arfken, G. (1985). *Mathematical Methods for Physicists.* 3rd edn. Academic Press.

Cho, T.S., Paris, S., Horn, B.K.P. and Freeman, W.T. (2011). Blur kernel estimation using the radon transform. *In IEEE Trans. Image Processing.*

Davis, L. (1975). A survey of edge detection techniques. *Computer Graphics and Image Processing*, vol. 4, pp. 248–260.

Fegus, R., Singh, B., Hertzmann, A., Roweis, S. and Freeman, W. (2006). Psf estimation using sharp edge prediction. *Removing Camera Shake from a Single Photograph*, vol. 27, pp. 787–794.

Fischler, M. and Bolles, R. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, vol. 24, pp. 381–395.

Gonzalez, R.C. and Woods, R.E. (2008). *Digital ImageProcessing.* Pearson Education Inc.

Helgason, S. (1999). *The Radon transform.* 2nd edn. Birkhauser.

Jahne, B., Scharr, H. and Korkel, S. (1999). *Principles of filter design. In Handbook of Computer Vision and Applications.* Academic Press.

Jia, J. (2007). Single image motion deblurring using transparency. *Proc. of Computer Vision and Pattern Recognition*, vol. 2, pp. 1–8.

Joshi, N., Szeliski, R. and Kriegman, D. (2008). Psf estimation using sharp edge prediction. *IEEE Conference on Computer Vision and Pattern Recognition.*

Kak, A.C. and Slaney, M. (2001). *Principles of Computerized Tomographic Imaging.* Society of Industrial and Applied Mathematics.

Kalke, M. and Siltanen, S. (2014). Sinogram interpolation method for sparse-angle tomography. *RSCIRP, Applied Mathematics*, vol. 5, pp. 432–441.

Levin, A. (2006). Blind motion deblurring using image statistics. *Advances in Neural Information Processing Systems*, vol. 2, pp. 1–8.

Levin, A., Weiss, Y., Durand, F. and T.Freeman, W. (2009). Understandingand evaluating blind deconvolution algorithms. *Proc. of Computer Vision and Pattern Recognition*.

Liu, C., Freeman, W.T., Szeliski, R. and Kang, S.B. (2006). Noise estimation from a single image. *CVPR '*06, vol. 2, pp. 901–908.

Lucy, L. (1974). An iterative technique for the rectification of observed distributions. *Journal of astronomy*, vol. 79, pp. 745–754.

Miskin, J. and Mackay, D.J.C. (2000). Ensemble learning for blind image separation and deconvolutione. *Adv. in Independent Component Analysis.*, vol. M. Girolani, Ed. Springer-Verlag.

Qi Shan, W.X. and Jia, J. (2007). Rotational motion deblurring of a rigid object from a single image. *Advances in Neural Information Processing Systems*.

Richardson, W.H. (1972). Bayesian-based iterative method of image restoration. *Journal of the Optical Society of America*, vol. 62, pp. 55–59.

Schuon, S. and Diepold, K. (2009). Comparison of motion deblur algorithms and real world deployment. *Paper on IAC*.

Stewart, J. (2012). *Calculus*. 7th edn. Brooks Cole.

Sun, H., Desvignes, M., Yan, Y. and Liu., W. (2009). Motion blur parameters identification from radon transform image gradients. *In Proc. of the conf. on Industrial Electronics*.

Toft, P. (1996). *The Radon Transform Theory and Implementation*. Ph.D. thesis, Technical University of Denmark, 2800 Lyngby. Denmark.

Weiner, N. (1949). Extrapolation, interpolation, and smoothing of stationary time series. *New York: Wiley*.